VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webový crawler

Web crawler

2011 Rostislav Zdeněk

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

V Ostravě 6. května 2011 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2011 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli. Mé poděkování patří především Mgr. Zdeňku Horákovi za ochotu a cenné rady při vedení mé bakalářské práce.

Abstrakt

Cílem této práce je vytvoření webového crawleru. Samotnému návrhu aplikace předchází analýza použití některých existujících řešení, popis jejich výhod, nevýhod a vlastností. Výsledkem by měla být volba vhodné technologie, návrh, realizace společně se specifikací některých problémů a představení vlastní implementace webového crawleru.

Klíčová slova: Internet, WWW, HTTP, URL, web, crawler

Abstract

Porpose of this labour is to create a web crawler. The suggestion of the application precedes an analysis of using some existing methods of solution, description of their advantages, disadvantages and qualities. The result of this analysis should be a choice of an acceptable technology, suggestion, realisation with specification of some problems and introduction of the web crawler's implementation.

Keywords: Internet, WWW, HTTP, URL, web, crawler

Seznam použitých zkratek a symbolů

OS	–	operační systém
HTTP	–	Hypertext Transfer Protocol (protokol aplikační vrstvy ISO/OSI modelu)
HTML	–	Hyper Text Markup Language (značkovací jazyk)
XML	–	Extensible Markup Language (rozšiřitelný značkovací jazyk)
GUI	–	grafické uživatelské rozhraní (graphics user interface)
CLI	–	rozhraní příkazového řádku (command line interface)
SSH	–	protokol přístupu ke vzdálenému systému (secure shell)
J2EE nebo JEE	–	platforma pro vývoj podnikových aplikací v prog. jazyku Java (Java 2 Enterprise Edition)
AS	–	aplikační server (prostředí pro provoz J2EE aplikací)
DB	–	databáze (databázový server)
ORM	–	objektově relační mapování (vrstva k propojení strukturovaných DB dat a obj. přístupu k programování)
REGEXP	–	regulární výraz (regular expression)
JVM	–	virtuální stroj java (java virtual machine)

Obsah

Seznam obrázků

1 Zadání

Cílem práce je vytvořit systém pro automatické stahování kolekcí webových stránek. Klíčové vlastnosti jsou:

1. Distribuovaná architektura umožňující rozložení požadavků mezi více instancí (ať už na jednom nebo více serverech).

2. Oddělení části stahující data od části plánující cíle stahování a administrace umožňující vkládání požadavků na stahování a zobrazení stavu plnění požadavku.

3. Součástí je zapouzdřený systém pro analýzu obsahu webových stránek generující seznam dalších požadavků ke stažení.

2 Úvod

Dnešní Internet si jen stěží dovedeme představit bez služeb umožňujících rychlé prohledávání obsahu, který je v něm jako celku dostupný. Schopnost efektivně vyhledávat informace na Internetu, patři k základním dovednostem dnešního člověka. Přední giganti softwarového průmyslu se stále předhánějí v tom, kdo nabídne lidem přesnější výsledky na jejich vyhledávací dotazy.

Na pozadí vyhledávacích služeb však stojí nutnost informace o obsahu Internetu nejprve nasbírat. Až následně je možné informace utřídit tak, aby mohly být s pomocí propracovaných matematických postupů rychle prohledávány a jako výsledky vyhledávání předkládány odkazy vedoucí zpět k dokumentům na Internetu, kde by tazatel měl najít odpovědi na své otázky.

Systematický *průchod* Internetu, přesněji řečeno sítě webových stránek, je pak úkolem programů, které jsou na tento úkol cíleně navrženy. Takové programy bývají obecně nazývány jako crawler, popř. je možné se setkat s označením robot, fetcher.

Tato práce se dá rozdělit do dvou částí: teoretické a praktické. Teoretická část si klade za cíl prozkoumat některá existující softwarová řešení, jejich použití, zamyslet se nad jejich výhodami, nevýhodami a specifiky. Výsledkem teoretické části by měla být představa konceptu pro praktickou část, kde se pokusím o realizaci a představení vlastní implementace webového crawleru.

3 Webový crawler

3.1 Pojem webového crawleru

V rámci tématu této bakalářské práce pod pojmem webový crawler rozumíme jisté softwarové řešení, sloužící pro systematické strojové procházení struktury webové prezentace, přičemž uvedené procházení webu má zpravidla nějaký další smysl. Crawler tedy *prochází* webové stránky a hledá cestu na další webové stránky, které si *zapamatuje* a dále v nich hledá cestu k dalším webovým stránkám...

Proces *průchodu* je jistě možné přirovnat k prohledávání adresářů a je zřejmé, že tento proces musí být vybaven jistou systematičností a pravidly.

3.2 Použití webového crawleru

Praktické důvody, které nás mohou vést k použití nějakého webového crawleru nebo snaze o vývoj vlastního řešení, mohou být různé. Uveďme jen namátkou:

- Můžeme potřebovat prověřit všechny odkazy v rámci domény a podchytit neplatné odkazy.
- Snaha zjistit početní, popř. objemové poměry v zastoupení různých druhů dokumentů, např. html, pdf, různé druhy obrázků a dalších.
- Sběr dat pro přípravu výtisku celého webu do jediného rozsáhlého dokumentu.
- Snaha projít celý vybraný web, popř. skupinu webů a hledat výskyt nějaké informace, např. chceme zjistit jestli na našem webu (např. podnikovém intranetu) někdo v nějaké části nepíše vulgarismy, popř. utajované skutečnosti a pod.
- Zatřídění a indexace obsahu k pozdějšímu efektivnímu prohledávání.

Zvláště pak poslední z uvedených možností nám již v mnohém připomíná součást moderních fulltextových vyhledávacích služeb. V rámci upřesnění je tedy nutné ještě jednou uvést, že samotný crawler je pouze prostředkem pro systematické procházení webu, přičemž další zpracování získaného obsahu je úkolem pro další software, resp. součásti spojené s crawlerem.

Je zřejmé, že v závislosti na tom, co od průzkumu vybrané části webu očekáváme, roste i složitost webového crawleru, který použijeme nebo se chystáme implementovat.

4 Analýza některých existujících řešení

Podívejme se před pokusem o návrh vlastní implementace webového crawleru na různá již existující řešení. Pro analýzu a možné srovnání byly vybrány různé softwarové produkty:

- Wget
- Teleport Pro
- Apache Nutch
- Google Mini

Zatímco program Wget se dá považovat za prostou a velmi čistou implementaci webového crawleru, tak protiváhou k tomu existují např. Apache Nutch nebo Google Mini jako komplexní softwarová řešení, obsahující také součásti pro indexování a prohledávání obsahu, který jejich crawler nasbírá.

4.1 Program Wget

Program Wget bývá většinou znám jako nástroj unixových a linuxových operačních systémů, kde se často používá k získání obsahu (webového dokumentu, obrázku, instalačního balíčku a obecně jakéhokoliv souboru) pomocí protokolu HTTP. Možné je však použít také protokoly HTTPS a FTP. Svým zpracováním se jedná o aplikaci pro příkazový řádek bez GUI. Často je také používán na pozadí různých nástrojů s GUI.

Méně známou vlastností programu Wget může být schopnost pracovat jako webový crlawler procházející HTML a XHTML stránky. V nich umí vyhledat odkazy na další dokumenty (soubory) a sekvenčně pak tyto vyhledané odkazy dále zpracovat, tzn. je stahovat a dále procházet je do dalších úrovní. Wget tedy umí provést rekurzivní stahování vybraného webu a tak např. umožňuje získat off-line kopii vybraného webu.

Součástí implementace Wgetu je také možnost řízení jeho práce podle jistých obecně uznávaných představ o slušném chování tohoto druhu software a chování podle, pravidel ze souborů robots.txt.

4.1.1 Instalace programu Wget

Program většinou bývá součástí instalace linuxových distribucí a unixů, popř. je možné jej snadno nainstalovat přes příslušný balíčkovací systém a nebo je možné program stáhnout v podobě přeloženého binárního souboru pro široké spektrum operačních systémů, včetně Windows. Program se řadí k tzv. opensource a dispozici jsou také zdrojové kódy. Wget je napsán v jazyku C. Web projektu je dostupný na http://www.gnu.org/software/wget/.

4.1.2 Práce s programem Wget

Nejjednodušším příkladem možného použití je provést *stažení* jediného dokumentu:

```
$ wget http://www.vsb.cz/
```

Výstup obsahu, který Wget získal z daného odkazu, se provede do souboru s názvem podle dokumentu na zadané URL, popř. jej můžeme určit parametrem -O. V tomto případě vznikne v aktuálním adresáři soubor index.html [1]

Mnohem zajímavější je použít Wget právě jako *procházeč* a *stahovač* různých částí a nebo rovnou celých struktur webů. Jedním ze základních způsobů takového použití je příkaz:

```
$ wget -p -k http://www.vsb.cz/
```

Výsledkem bude vytvoření adresáře ./www.vsb.cz , kam se uloží stažený dokument, tj. index.html. Zadání obsahuje také požadavek na stažení dalších součástí, které jsou potřeba pro úplné zobrazení stránky v prohlížeči (parametr -p) a konverze (parametr -k) jednotlivých odkazů v získaném dokumentu (index.html) tak, aby tyto odkazy vedly na lokální úložiště staženého obsahu (styly, obrázky a pod.) a umožnily stránku zobrazit i bez připojení k síti.

Pokračujme ukázkou použití rekurze při zpracování jednotlivých získaných odkazů z dokumentu:

```
$ wget -r -l 5 http://www.vsb.cz/
```

Příkaz znamená, že se začne na dokumentu (bude jím bude výchozí dokument index.html) z http://www.vsb.cz a dokument bude dále rekurzivně (parametr -r) zpracováván. Rekurzivního zanoření je zde určeno (parametr -l) max. až do 5. úrovně.

Další významnou vlastností programu Wget je možnost práce s hodnotami cookies a automatické vyplňování webových formulářů. Máme-li tedy např. nějakou oblast webu, která je chráněna jménem a heslem, tak je možné nechat Wget automaticky přihlásit. Informace o přihlášení jsou často udržovány s pomocí cookies, které lze s Wgetem přijímat a zpětně zase vracet a takto lze procházet i části webu, které se na první pohled zdají strojově nezpracovatelné. Zde je dvojice příkazů:

```
 $ wget --no-check-certificate
--secure-protocol=SSLv3
--keep-session-cookies
--save-cookies=cookies.text
--post-data "login=$LOGIN&passwd=$PASSWD"
"https://www.nejakyweb.cz/login.php"

 $ wget --no-check-certificate
--secure-protocol=SSLv3
--keep-session-cookies
--load-cookies=cookies.text
"https://www.nejakyweb.cz/obsah.php"
```

[1] Protože v příkazu chybí v URL uveden konečný dokument, webový server vrátí (podle svého nastavení) výchozí dokument, zde je to index.html.

Uvedené příklady představují použití Wgetu přes protokol HTTPS (neověřuje pravost šifrovacích certifikátů) pro automatické přihlášení. Odeslání formuláře je provedeno metodou POST. Řetězce login a passwd představují pojmenování formulářových políček pro vyplnění jména a hesla, řetězce $LOGIN a $PASSWD jsou proměnnými shellu, které nastavíme potřebnými informacemi. Vidíme také parametry pro práci s cookies, kde v prvním příkazu cookie získáme a ve druhém příkazu ji předáváme zpět.

Práce programu Wget se za jistých okolností může stát pro cílový web značně nepříjemná. Může docházet k nadměrnému zatížení webového serveru a následnému zhoršení odezvy a spolehlivosti při stahování obsahu, také může dojít k zabrání šířky síťové konektivity. Tak si zhoršíme podmínky sami a nebo na základě našeho chovaní dojde k omezení dostupnosti webu pro běžné uživatele. Není-li web, který procházíme náš, pak často také následuje odezva ve formě omezení našeho přístupu ze strany správce v podobě (dočasného) zablokování IP adresy. Wget proto obsahuje také možnosti nastavení pro své chování při provádění úkolu.

```
wget -r -l 100 -w 3 "http://www.vsb.cz"
```

Závěrečný příklad by znamenal rekurzivní průchod vybraným webem až do sté úrovně. Mezi jednotlivými následujícími dotazy by se ponechávala náhodně dlouhá pauza v rozmezí od 0 do 3 sekund s jemností na milisekundy.

Ve výchozím nastavení (spuštění) Wgetu je zahrnuto chování podle robots.txt. Chování se obecného crawleru podle tohoto souboru patří (společně např. s výše zmíněným nezahlcením serveru) opět k jisté *slušnosti*, která by bezdůvodně neměla být přehlížena. Wget tedy soubor robots.txt hledá a dle něj se také na prohledávaném webu chová. Tuto vlastnost můžeme potlačit pomocí zadání `$ wget -e robots=off ...` . Wget pak bude robots.txt ignorovat.

Z mnoha důvodů (specifický obsah pro různé prohlížeče, omezení přístupu, ...) můžeme chtít změnit HTTP hlavičky [2], které Wget odesílá. Tato možnost je přístupná pomocí zadání `wget -U "Mozilla/5.0 (compatible; Konqueror/3.2; Linux)"...`. Cílový webový server si tak bude myslet, že Wget je běžný Firefox.

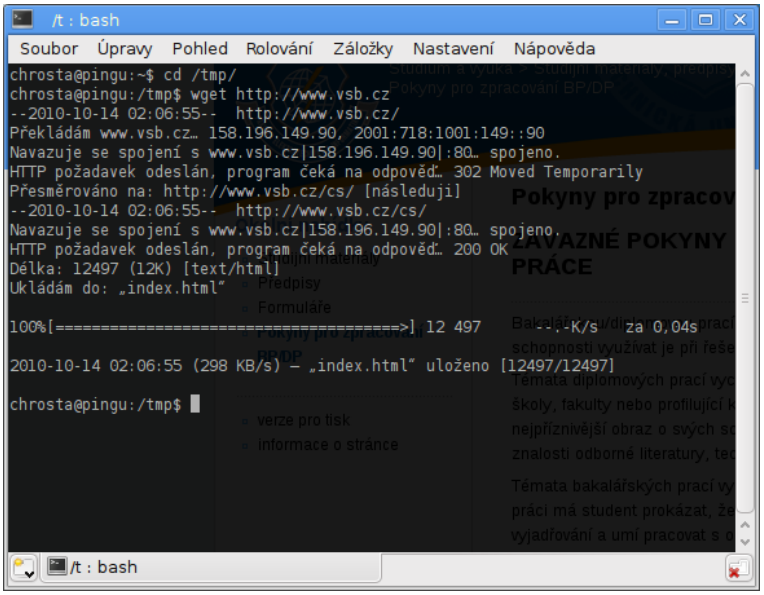
Je toho skutečně mnoho, co Wget umí a co je v něm možné nastavit...

4.1.3 Shrnutí poznatků a práce s programem Wget

Program Wget (obrázek 1) je bezesporu velice mocným nástrojem, přestože na první pohled vypadá jen jako malá utilita. Při práci s ním se však můžeme setkat s některými vlastnostmi, které mohou znamenat jisté omezení:

Program musí zadanou úlohu zpracovat na jedno spuštění. Pokud běh programu přerušíme, rozpracovanou úlohu již nelze jednoduše obnovit a pokračovat v ní dále. Významná je vlastnost sekvenčního zpracování nalezených odkazů a jednoprocesové řešení. Prostým spuštěním programu nedocílíme například situace, kdy v systému mohou

[2] Vnitřní součást HTTP protokolu, kterou běný uživatel nemusí nijak vnímat a která obsahuje nejrůznější řídící informace.



Obrázek 1: Ukázka programu Wget při stažení úvodní stránky VŠB-TUO.

pracovat dvě vlákna Wgetu v rámci stejné úlohy. V době vícejádrových a více vláknových procesorů to může být škoda. Podobně se může vicejádrová a vícevláknová architektura hodit v situacích s vysokou kapacitou / propustností připojení, kdy jedno vlákno stahuje kompletní obsah pro zobrazení jedné stránky, ale linka zůstává stále z velké části nevyužita a těchto stahování by mohlo ve stejné chvíli běžet více. Některým uživatelům nemusí vyhovovat uživatelské rozhraní ve formě příkazové řádky.

Zůstává však faktem, že jen málo podobných (ve zdánlivé podobě utility) programů může nabídnou tak propracovanou podporu pro další aplikační řešení nejrůznějšího účelu. Wget například slouží v programu HTTrack (http://www.httrack.com/) , který je podobný dalšímu programu Teleport. V HTTracku je řešeno GUI, vláknování a navazování práce, přičemž vše stojí na kvalitě Wgetu.

4.2 Program Teleport Pro

Program Teleport je dílem společnosti TENMAX a je to placený uzavřený software, který je určen především k vytváření kopií webových prezentací. Je vybaven GUI, přes které lze nastavit projekt ke stahování a dále obsahuje možnost spustit minimalizovaně program, který se sterá o stahování aniž by musel být spuštěn program s hlavním oknem aplikace. Teleport je dostupný na http://www.tenmax.com/teleport/pro/home.htm. Program byl testován ve zkušební licenci, ve verzi 1.62 a je základní verzí programu Teleport, přičemž verze Pro je následována verzemi Ultra, VLX, Exec a Exec/VLX. Dále k verzi Teleport Pro výrobce uvádí několik vlastností:

- Spuštění až 10 souběžných vláken stahujících webový obsah.

- Stažení všech součástí (souborů) webu a jejich uspořádání, tak, že je pak stažený web možno procházet jen z disku, což je pak nepoměrně rychlejší.
- Vytvoření přesné kopie (zrcadla) webu s adresářovou strukturou odpovídající uspořádání zrcadleného webu.
- Prohledávání cílového webu na soubory o dané velikosti a typu.
- Možnost přístupu na stránky chráněné jménem a heslem.
- Získání seznamu souborů na dané adrese.
- Průzkum webů, které jsou odkazovány ze zadaného počátečního zkoumaného webu.
- Prohledávání zadaného webu na výskyt určitých slov.
- Vytvoření seznamu všech souborů a stránek z daného zkoumaného webu.

4.2.1 Instalace programu Teleport Pro

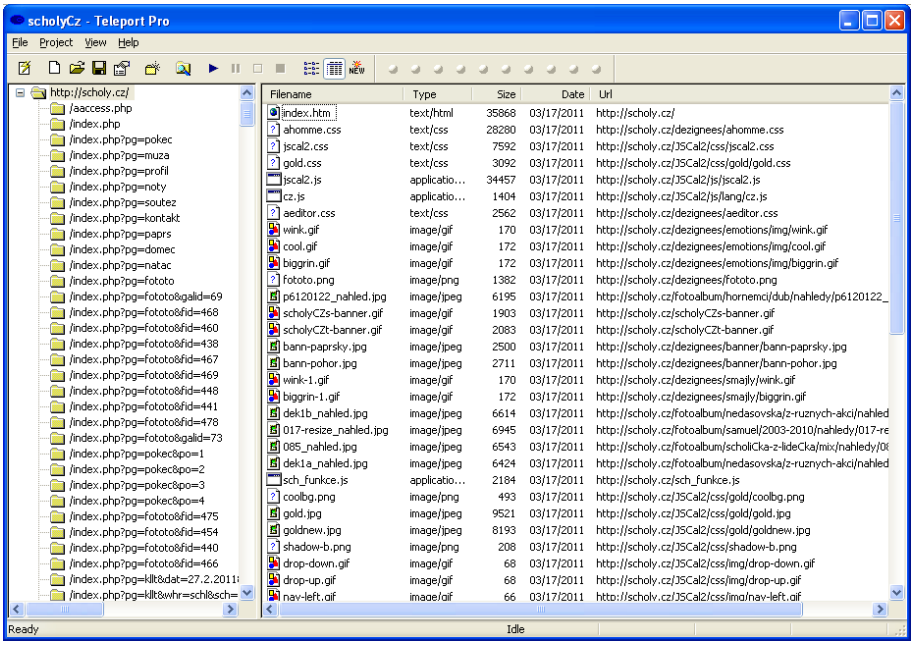
Instalace programu Teleport Pro je prostá a neliší od běžného způsobu instalace programů v OS Windows. Ze stránek získáme instalační binárku, tu dle zvyklostí ve Windows jednoduše nainstalujeme a spouštěče programu pak najdeme v příslušné nabídce systému. Je zde spouštěč pro hlavní část programu s manažerem, kde se definují úlohy a pak již zmíněný program *Scheduler* pro oddělené provádění naplánovaných úloh.

4.2.2 Použití programu Teleport Pro

Spuštění programu je prosté, volíme Teleport Pro v příslušné nabídce spouštěčů. Přivítá nás vcelku jednoduché a intuitivní GUI, kde si ihned můžeme všimnout nástrojové části panelu a deseti (šedých / neaktivních) indikátorů stavu stahovacích vláken. Na úvodním obrázku je již načten také projekt scholyCz pro web http://scholy.cz, na kterém jsem program Teleport testoval.

Pokusme se prozkoumat vlastnosti programu Teleport Pro založením vlastního projektu a pak dále analyzovat práci Teleportu na tomto projektu. Vytvoření nového projektu provedeme pomocí integrovaného průvodce, který nás provede řadou 4 kroků:

1. V prvním kroku zadáme co přesně chceme s pomocí Teleportu dělat. Na výběr máme:
   (a) Vytvoření průchozí kopie zadaného webu na lokálním disku. (Připojíme-li do systému např. vzdálený síťový disk, lze samozřejmě použít i tento.)
   (b) Kopie zadaného webu tak, že je kopírována také adresářová struktura. (V předchozí volbě TeleportPro ukládá stažené stránky a jejich zdroje do adresáře projektu, soubory od sebe odlišují číselné haše.)
   (c) Prohledávání zadaného webu na vybrané typy souborů.



Obrázek 2: Základní obrazovka Teleport Pro s projektem stažení http://scholy.cz .

(d) Průzkum každého webu, na který je odkazovaného ze zadaného / centrálního webu.

(e) Nahrazení jednoho nebo více souborů za známou adresu.

(f) Prohledání zadaného webu na vybraná klíčová slova.

2. Ve druhém kroku je potřeba zadat:

   - Počáteční adresu, od které se s průchodem webu začne.
   - Hloubku zanoření od počáteční adresy do logické struktury webu. Výchozí hodnota je 3, ale použijeme hodnotu 5.

3. Třetí krok vyžaduje definování dat, která chceme stahovat a kopírovat na lokální disk:

   - Jen textová data. (HTML, styly, doplňující textové soubory a pod.)
   - Textová data a grafiku. (K textovým datům budeme stahovat také obrázky, tj. JPEG, GIF, PNG a pod.)
   - Testová data, grafiku a zvuky. (K již uvedením se přidávají audio data, tj. WAV, MP3 a pod.)
   - Všechno. (K výše uvedenému budou stahovány i další soubory, tj. Java Applety, ZIP archívy a další.)

Obrázek 3: Ukázka indikátoru stavu stahovacích vláken v Teleportu při práci na projektu.

Obrázek 4: Dokončení zpracování všech nalezených odkazů Teleport Pro oznámí...

Jestliže je web chráněn HTTP autentizací, máme zde možnost zadat platný účet a heslo, které se použije pro přihlášení.

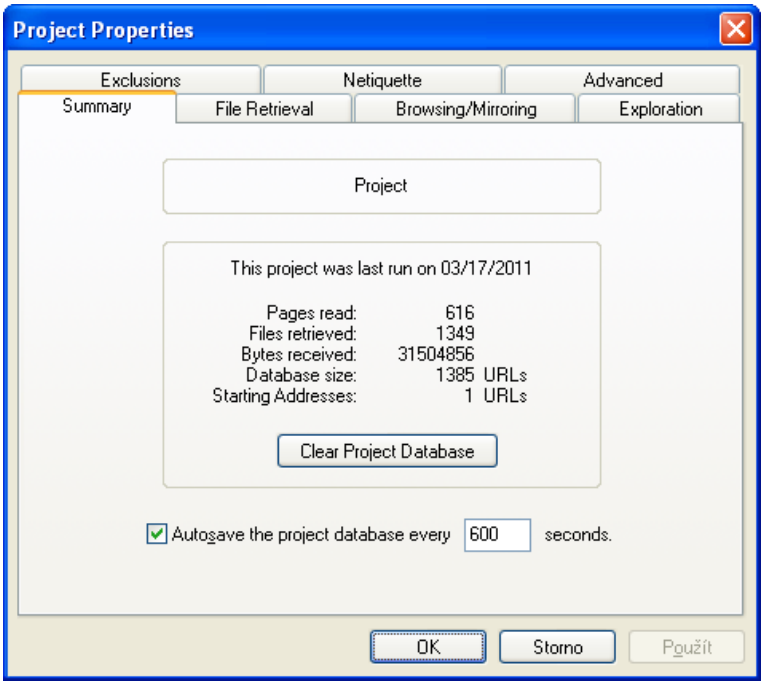
4. Ve čtvrtém kroku je zadání nového projektu kompletní, projekt uložíme a tlačítkem *start* jej můžeme ihned nechat provádět.

   V programu nechybí možnosti nastavení připojení k internetu, především proxy, další možnosti nastavení projektu a chovaní můžeme provádět v nabídce *Project→Project properties*, odkud zmiňme:

   - Statistika projektu.
   - Pravidla pro stahování souborů, typ, velikostní omezení...
   - Pravidla pro ukládání souborů, lokalizaci (ve smyslu úmístění na místním disku) odkazů...
   - Pravidla pro průzkum, opakování a chování se v případě neuspěšných a chybových pokusů, počet souběžných stahovacích vláken, práce s cookies...
   - Aktualizace stažených dat po zadané době.
   - Nastavení programu *Scheduler*, automatické a obnovené spuštění projektu...
   - Nastavení výjimek pro soubory, které nechceme procházet.
   - Netiquette, nastaveni pravidel internetové etikety, pro *slušné* chování crawleru Teleportu, tj. prodlevy mezi jednotlivými HTTP požadavky, identifikace, hlídání odezev stahovaného webu a přizpůsobení počtu stahovacích vláken...

Uvedený proces vytváření projektu v Teleportu nám dává poměrně dobrý přehled o vlastnostech, parametrech práce a množství problémů, které může být potřeba při práci (a v neposlední řadě i vývoji) webového crawleru určit a vyřešit.

Necháme-li projekt spustit, aplikace založí ve vybrané složce adresář projektu a začne stahovat vybraný web. Na zmíněném indikátoru stavu stahovacích vláken můžeme sledovat průběh akcí. Ve stavovém řádku hlavního okna je stále aktualizován počet nalezených odkazů, dále počet odkazů, které už byly zpracovány a s tím související počet odkazů, které ještě zbývá zpracovat. Nechybí také informace o objemu přenesených dat. To nám dává dobrý přehled o rozsahu vykonané a zbývající práce Teleportu.



Obrázek 5: Statistika projektu s počtem stránek (616) a počtem souborů (1349).

Pro otevřený projekt si můžeme nechat zobrazit statistiku, která obsahuje především informace o přečtených stránkách a celkovém počtu stažených souborů. Tato dvě čísla použijeme při srovnání s dalšími programy.

4.2.3 Shrnutí poznatků a práce s programem Teleport Pro

Program TeleportPro je bezpochyby kvalitní software. Mezi pozitiva a výhody je možné zařadit nabídku množství užitečných funkcí a možností nastavení práce crawleru, přesto si program zachovává značnou jednoduchost a přehlednost, která jej činí velmi dobře použitelným. Jako jisté negativum je snad možné zmínit závislost na platformě OS Windows, což může být svazující. Přes všechnu funkční pestrost se však program zdá určen spíše pro desktopové použití a prosté stahování, popř. zálohování webů. Samozřejmě zůstává možnost použít nebo vytvořit další specializované programy či skripty ke zpracování stažených dat. Zamyslíme-li nad možnostmi škálování, připadá v úvahu jen škálování vertikální. Možnosti práce více programů Teleport na více počítačích nad stejným úkolem je značně omezená. Teleport Pro nepracuje se soubory robots.txt.

V případě pokročilejších verzí programu (Ultra, VLX, Exec, Exec/VLX) jsou implementovány další pokročilé vlastnosti pro použití Teleportu. Verze Ultra nabízí např. použití regulárních výrazů pro specifikaci odkazů k zahrnutí do (nebo vyloučení) z procházení, podpora HTTPS při stahování, vkládání vlastních HTTP hlaviček do požadavků. Verze VLX pak rozšiřuje možnosti Teleportu podporou pro rozsáhlé kolekce až 40 mil.

odkazů na projekt, rozložení práce na jednom projektu mezi více počítačů. Verze Exec je vybavena rozhraním pro použití ve vlastních projektech, popř. nabízí možnosti analýzy dynamického HTML k průchodu i velmi složitých webů. Verze Exec/VLX je kombinací nejpokročilejší verze Exec s možností rozsáhlých kolekcí verze VLX.

4.3 Projekt Apache Nutch

Projekt Apache Nutch je již komplexním řešením, které dohromady zahrnuje vyhledávací stroj (search engine a nástroj pro indexaci) a vlastní webový crawler, který je napsán od základu a zvláště pro tento projekt. Celé řešení se zdá již od pohledu cíleno na serverová využití v podnikových sítích, sběr rozsáhlých dat, jejich možnou analýzu a podobně.

Nutch je napsán v programovacím jazyku Java. Projekt je navržen s ohledem na vysokou modulárnost, která umožňuje vývojářům (projekt je opensource) vytvářet zásuvné moduly a dále tak rozšiřovat funkčnost v získávání, zpracování a řešení dotazování dat. Podstatnou vlastností je vysoká škálovatelnost celého řešení. [6] Pro efektivní škálování je v projektu využíván zvláštní framework pro spolehlivé, škálovatelné a distribuované výpočty Apache Hadoop. Nasbíraná data jsou ukládána *jazykově* (na Javě) nezávisle, čímž jejich další zpracování není nijak omezeno.

Od roku 2010 je Nutch zahrnut pod Apache Software Foundation, kde je spravován jako jeden z hlavních projektů a podporován např. společnostmi Yahoo!, popř. The Internet Archive.

4.3.1 Instalace projektu Apache Nutch

Současná nejnovější verze Apache Nutch je 1.2 ze září 2010. Jedná se o Java aplikaci vyžadující nainstalované JDK a aplikační server, zde dokumentace uvádí Apache Tomcat 6.x. Web projektu je dostupný na http://nutch.apache.org/about.html.

4.3.1.1 Instalace JDK a aplikačního serveru Tomcat

Instalace JDK spočívá ve stažení příslušného binárního balíčku ze http://java.com, rozbalení balíčku a nastavení proměnných prostředí $PATH a $JAVA_HOME. K instalaci Apache Tomcat opět získáme z http://tomcat.apache.org/ binární balíček [3], který rozbalíme do vhodného místa našeho filesystému. Aplikační server spustíme příslušným startovacím skriptem `/opt/tomcat6/bin/startup.sh` a funkčnost aplikačního serveru ověříme v prohlížeči na http://localhost:8080 zobrazením uvítací stránky Tomcatu.

Doplňme skutečnost, že Tomcat bude sloužit pro přístup k prohledávání nasbíraných webových stránek, resp. bude sloužit k provozu vyhledávací aplikace balíčku Nutch. (Balíček obsahuje potřebný aplikační archív WAR.) Crawler, který řeší prohledávání webu, je další (a v rámci této bakalářské práce zásadnější) částí balíčku Nutch, ale tento crawler nijak nesouvisí s Tomcatem. (Webové stránky můžeme nasbírat i bez Tomcatu.) S prostředky balíčku Nutch dále zvládneme indexování nasbíraných dat i jejich prohledávání z prostředí příkazové řádky.

[3] Pro účely testovaní byl použit balíček apache-tomcat-6.0.32, JDK 1.6.0 update 21 a testování bylo prováděno na Linuxu.

4.3.1.2 Instalace Nutch

Ze stránek projektu si stáhneme připravený binární instalační balíček, který jen rozbalíme do námi vybraného místa. Nastavíme proměnné prostředí $NUTCH_JAVA_HOME (můžeme použít proměnnou $JAVA_HOME) a $NUTCH_HOME podle místa rozbalení balíčku.

Po instalaci je vhodné provést základní nastavení v souboru `nutch-default.xml` , kde nastavíme jméno crawleru v proměnné `http.agent.name` , můžeme doplnit `http.agent.url` a `http.agent.email` . Tyto proměnné se použijí k identifikaci našeho crawleru při přístupu na prohledávaný web a odpovídají hlavičce, kterou typicky předkládají webové prohlížeče. Nyní mějme Nutch naistalován a nastaven v adresáři `/opt/nutch12` .

4.3.2 Práce s Apache Nutch

Založení stahovacího úkolu (ve smyslu projektů, které jsme poznali v programu Teleport Pro) a spuštění crawleru Nutche, se provede několika snadnými operacemi... Postup by měl zohlednit i to, že úkolů můžeme chtít několik vedle sebe pro více než jeden web.

4.3.2.1 Vytvoření nového úkolu pro crawler Nutch

V adresáři s Nutch si vytvoříme adresář `crawls` , kde si dále vytvoříme textový soubor `urls` pro *startovací* odkazy úkolů a adresář `scholy` pro konkrétní úkol a jeho data. Do `urls` přídáme první (*startovací*) odkaz `http://scholy.cz` .

4.3.2.2 Konfigurace crawleru Nutch

Pokračujeme úpravou konfiguračního souboru `crawl-urlfilter.txt` , kde nastavíme doménové jméno a počáteční URL webu, který chceme procházet, takže nahradíme `MY.DOMAIN.NAME` za `scholy.cz` . Soubor definuje pravidla, podle kterých budou na prohledávaném webu rozeznávány další odkazy a jak tyto odkazy budou vyhodnocovány, tj. jestli budou zařazeny nebo ignorovány pro další prohledávání. Přesná podoba konfiguračního souboru je tak značně závislá na tom, co přesně od crawleru Nutch požadujeme. Pro potřebu testování jsem tedy zvolit následující nastavení:

```
# skip file:, ftp:, & mailto: urls
-^(file|ftp|mailto):
```

Nastavení zajistí ignorování odkazů začínajících na vybrané řetězce.

```
# skip image and other suffixes we can't yet parse
-\.(js|JS|gif|GIF|jpg|JPG|png|PNG|ico|ICO|css|sit|eps|wmf|zip|
    ppt|mpg|xls|gz|rpm|tgz|mov|MOV|exe|jpeg|JPEG|bmp|BMP)$
```

Nastavení zajistí ignorování odkazů končících na vybrané řetězce. Oproti výchozímu nastavení jsem zde dodal ještě "js"a "JS".

```
# skip URLs containing certain characters as probable queries, etc.
# -[?*!@=]
```

Nastavení zajistí přeskočení odkazů, které obsahují alespoň jeden z uvedených znaků. Původně výchozí nastavení jsem však zakomentoval a není použito, protože cílový web, pracuje se značným množstvím dynamicky generovaného obsahu a odkazy obsahují argumenty s hodnotami předávanými pomocí metody GET.

```
# skip URLs with slash-delimited segment that repeats 3+ times,
# to break loops
-.*(/[^/]+)/[^/]+\1/[^/]+\1/
```

Užitečné nastavení, které brání zacyklení crawleru na odkazech.

```
# accept hosts in MY.DOMAIN.NAME
+^http://([a-z0-9]*\.)*scholy.cz/
```

Zde nastavíme vyhovující doménu, zpravidla doménu k prohledávání, kde chceme crawler *udržet* . Pokud chceme procházet více domén, přidáme další řádek.

```
# skip everything else
-.
```

Poslední pravidlo v souboru, které zajistí ignorování všeho ostatního.

Každé z uvedených nastavení, resp. pravidel, má znaménko. Celý konfigurační soubor `crawl-urlfilter.xml` funguje tak, že po vyparsování odkazu na stránce je tento odkaz podroben testu na jednotlivých pravidlech podle pořadí od shora dolů. Testování končí na prvním pravidlu, regulárním výrazu, kterému odkaz vyhoví a nad odkazem se podle znaménka provede rozhodnutí o jeho zahrnutí do dalšího plánu (+) a nebo jeho ignorování (-). Popsané řešení nabízí elegatní a jednoduchou logiku, která umožňuje i velmi jemné nastavení práce crawleru.

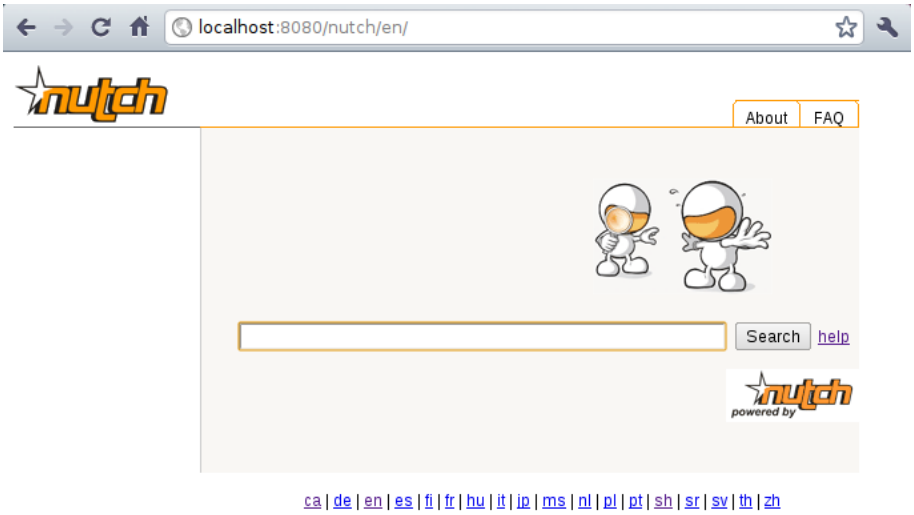
Všechna nastavení Nutche jsou dostupná v adresáři `/opt/nutch12/conf` . Názvy jednotlivých konfiguračních souborů poměrně jasně naznačují, se kterou komponentou nebo vlastností Nutche souvisejí a jsou dobře okomentovány. Nechybí samozřejmě nastavení pro HTTP proxy, chování vůči cílovému webovému serveru, omezení proti stahování dat přesahujících volitelnou velikost a další možnosti.

4.3.2.3 Spuštění crawleru Nutch

Když máme vytvořený úkol a nastaveny parametry práce crawleru, můžeme spustit samotný crawler:

```
$ cd /opt/nutch12/crawls
$ ../bin/nutch crawl urls -dir ./scholy -threads 3 -depth 5
```

Význam jednotlivých parametrů je poměrně zřejmý: Žádáme spuštění crawleru Nutche (`crawl`) pro URL v `urls` a výsledky práce (nasbíraný obsah, odkazy, indexy, ...) chceme ukládat do `./scholy`, k práci mají být použita tři vlákna (`threads`) a prohledávání má probíhat až do páté úrovně (`depth`). O průběhu práce Nutche jsme informování přehlednými výpisy. Zpracování testovacího úkolu bylo provedeno za 3 min. a 48 sek.



Obrázek 6: Přístup k webové aplikaci výhledávače Nutch.

4.3.2.4 Analýza dat z Nutche

Výsledky práce crawleru Nutch jsou umístěny do adresáře `./crawls/scholy` , kde najdeme několik podadresářů:

- `crawldb` obsahuje informace o každé URL nalezené Nutchem, zda byla URL prozkoumána a kdy naposled se tak stalo.
- `linkdb` obsahuje seznam všech odkazů na každou URL, zahrnuje společné zdroje jednotlivých URL a *kotvy* odkazů.
- `segments` obsahuje sady segmentů. Každý segment (časovým razítkem očíslovaný podadresář) je výsledkem zpracování sady URL a obsahuje podadresáře:
  - `content` obsahuje *syrová* data stažená z každé URL.
  - `crawl_fetch` obsahuje status každé URL.
  - `crawl_generate` obsahuje jména sad URL k dalšímu prozkoumání.
  - `crawl_parse` obsahuje oddělené URL používané pro `crawldb` .
  - `parse_text` obsahuje rozparsovaný text z každé URL.
  - `parse_data` obsahuje odkazy na data a metadata pro každou URL z prozkoumané URL.
- `index` a `indexes` obsahují indexy nad nasbíranými datovými segmenty.

Tento celek tvoří vlastní databázi úkolu. Nutch při své práci průběžně obnovuje obsah této databáze, třídí jednotlivé odkazy, parsuje a indexuje nasbíraný obsah. Výslednou databázi pak můžeme prohledávat...

4.3.2.5 Prohledávání databáze Nutche

Nejsnadnější způsob, jak ověřit integritu a použitelnost nasbíraných dat, je zkusit na datech prohledávání. K tomu použijeme vnitřní součást NutchBean. Než začneme výsledky prohledávat, **je důležité uvést, že NutchBean vždy očekává nasbíraná data (databázi) v adresáři `./crawl`**! V adresáři s úkoly můžeme vytvořit symbolický link:

```
$ cd /opt/nutch12/crawls
$ ln -s ./scholy crawl
```

Samotné vyhledávání provedeme pomocí příkazu:

```
$ /opt/nutch12/bin/nutch org.apache.nutch.searcher.NutchBean team
```

Příkazem spustíme java beanu NutchBean a necháme vyhledávat slovo "team". Výsledkem hledání bude výpis:

```
Total hits: 725
0 20110319132357/http://adorare.scholy.cz/index.php?pg=scholas&sch=5
... & design by scholy.cz team
1 20110319132357/http://ag.scholy.cz/index.php?pg=scholas&sch=12
... & design by scholy.cz team
2 20110319132357/http://bozibang.scholy.cz/index.php?pg=scholas&sch=6
... & design by scholy.cz team
3 20110319132357/http://delfini.scholy.cz/index.php?pg=scholas&sch=7
... & design by scholy.cz team
```

Vidíme, že v nasbíraných datech se nalezlo celkem 725 odkazů, které vedou ke slovu "team". Je dobré mít alespoň hrubou představu o očekávaných výsledcích hledání. Uvažujme, že prakticky na každé stránce testovacího webu nalezneme v patičce slovo "team"a pokud provedeme srovnání s množstvím stránek nalezených Teleportem Pro (616), zjistíme, že si čísla zhruba odpovídají. Můžeme tak předpokládat, že základní nastavení crawleru Nutch se podařilo.

4.3.2.6 Apache Nutch jako webový vyhledávač

Jak již bylo zmíněno, Nutch obsahuje také webovou aplikaci pro prohledávání, kterou si můžeme nainstalovat do Tomcatu. V adresáři `/opt/tomcat6/webapps` si vytvoříme aplikační adresář `nutch` , kde nakopírujeme soubor `nutch-1.2.war` a rozbalíme jej pomocí `jar -xvf nutch-1.2.war` . Soubor pak můžeme vymazat. Vyhledávací aplikace také využívá komponentu NutchBean a tak v adresáři aplikace musíme zajistit přítomnost adresáře `./crawl`. To v aplikačním adresáři snadno zařídíme pomocí symbolického linku do adresáře Nutche `/opt/nutch12/crawls/crawl`.

V adresáři (**to je důležité!**) `/opt/tomcat6/webapps/nutch` spustíme Tomcat pomocí `/opt/tomcat6/bin/catalina.sh start`.

Připojíme se pomocí webového prohlížeče na `http://localhost:8080/nutch/en/` a vyzkoušíme vyhledávání... Na závěr pak stačí vyhledávací aplikaci jen ostylovat a můžeme mít *vlastní* vyhledávač např. v podnikovém intranetu.

Obrázek 7: Vyhledávání pomocí webové aplikache Nutche.

4.3.3 Shrnutí poznatků a práce s projektem Nutch

Použití projektu Nutch se na první pohled může zdát složité, ale jedná se o velice povedený projekt. V jednom balíku získáváme crawler společně s vlastním řešením ukládání nasbíraných dat, nad kterým funguje indexace a vyhledávání. Součástí vyhledávání je možné data nechat zobrazit z cache.

Aplikace zcela jistě přesahuje rámec běžného uživatelského desktopu a je určena spíše pro serverová řešení, přitom však základní konfigurace jednotlivých částí do funkčního celku zůstává velmi jednoduchá.

4.4 Řešení Google Mini

Google Mini je okrajovým a posledním zástupcem v tomto přehledu některých existujích řešení crawlerů. Jedná se o řešení dodávané společností Google, resp. jejím obchodním

Obrázek 8: Zařízení Google Mini dodávané ve formě 1U boxu k instalaci do racku. [8]

zastoupením v konkrétní zemi. Samotná dodávka je provedena ve formě hardware, kterým je server ve formátu 1U připravený k okamžitému zamontování do racku.

Google Mini tedy přestavuje možnost jak získat technologii vyhledávače Google do firemního intranetu a využít některé speciální vlastnosti a schopnost crawleru. Především jde o schopnost procházet a indexovat nejrůznější soubory. Systém může procházet např. PDF soubory a soubory ODF z kancelářského balíku Open Office, nechybí podpora pro archívy ZIP, RAR a pod. a součástí řešení je také zpracování uzavřených formátů, typicky souborů DOC, XLS, PPS a dalších z kancelářského balíku Microsoft Office.

K zařízení je poskytována odpovídající podpora a závady jsou řešeny výměnou hardware. Přístup k nastavení, kontrolování stavu a řízení práce celého systému je poskytován pouze formou webového rozhraní, kde existuje administrátorský účet a účet uživatele manažer. Další přístup k zařízení zákazník nemá a tak není např. možné se ke Google Mini a jeho operačnímu systému připojit pomocí SSH. Přes webové rozhraní probíhají také aktualizace softwaru. Dále je možné nastavit, aby zařízení o své práci a svém technickém stavu podávalo pravidelné reporty formou jednoduchého emailu na vybranou adresu. Přes webové rozhraní lze přistupovat také k logům webového serveru Google Mini a ke statistikám vyhledávaných výrazů.

Nastavení zařízení v prostředí webového rozhraní je velmi snadné. Základem je startovací odkaz, kde se má začít indexovat. Další nezbytnou věcí je nastavení šablony pro vyhovující odkazy k dalšímu procházení a stejně tak šablony pro odkazy, které se dále nemají procházet a nemají být indexovány. Tyto podmínky pro procházení se nastaví ve formě regulární výrazů. Při své práci Google Mini akceptuje soubory `robots.txt` a metatagy v hlavičkách HTML stránek.

Google Mini je možné v intranetu využít v stejným způsobem jako vyhledávač Google na Internetu, tj. připojíme se k serveru a do vyhledávacího políčka napíšeme hledaný výraz, a nebo je možné jej využít jako vyhledávací platformu pro další aplikaci, kdy jsou výsledky vyhledávání vráceny ve formě XML, které může aplikace snadno prezentovat svým vlastním způsobem nebo jej dále jinak vyžít. Cena dodávky se odvíjí od množství dokumentů, tj. rozsáhlosti webu, které budeme chtít indexovat.

4.5 Shrnutí analýzy existujících řešení

Prozkoumali jsme některé existující implementace webových crawlerů. Každé řešení bylo vyzkoušeno, což vyžadovalo získání instalačních balíčků, instalaci, seznámení se s základní konfigurací, ovládáním a použitím. Jednotlivá prezentovaná řešení byla vybrána s ohledem na jejich určitou různorodost a pestrost.

Existuje mnoho programů v podobě Wgetu, jakožto zástupce spíše utilitek. Ovšem propracovanost Wgetu jej z oblasti jednoduchých utilit jistě lehce odsouvá k pokročilejším programům. Díky jeho implementaci v C můžeme očekávat i značné výkonnostní schopnosti a možnost program přeložit pro široké spektrum OS. Instalace programu spočívá jen (pomineme-li vlastní překlad) v jeho umístění na disk a hned jej můžeme používat. Díky CLI je však jeho použití spíše pro šikovnější počítačové uživatele.

Teleport je zastupcem desktopových řešení. Program má příjemné GUI a různá nastavení. Takové programy většinou umí stáhnout vybraný web, který je pak možné prohlížet z lokálního umístění bez připojení k internetu. Využití takových programů však samozřejmě nemusí končit u prostého kopírování webu. Získanou kopii můžeme dále zpracovat. Instalátor program nainstaluje do OS. GUI umožňuje program ovládat i méně zkušeným uživatelům, popř. je vede spíše ke studiu nastavované vlastnosti, než ke studiu jak danou vlastnost nastavit.

Apache Nutch je určen jistě spíše na servery. Při jeho návrhu byl brán ohled na mnoho důležitých vlastností jako je výkon a možnosti škálování. Implementace v programovacím jazyku Java umožňuje jeho použití na různých OS. Instalace, základní konfigurace a použití by pro šikovnějšího uživatele (např. firemního administrátora) nemělo být vážnějším problémem.

Google Mini představuje možnost jak se zcela oprostit od nutnosti se s něčím *trápit*. Je nám dodáno zařízení, které velmi jednoduše a s minimálními znalostmi můžeme nastavit a které začne pracovat. Jednoduchost, ale vnímejme i jistou profesionalitu, takového řešení však něco stojí.

5 Implementace vlastního webového crawleru

Následující část této práce je věnována implementaci vlastního řešení webového crawleru. Postupně probereme nutné vlastnosti, představu výsledku, výběr technologie, postup implementace, představení produktu, jeho instalace a použití.

5.1 Nutné vlastnosti crawleru

Z předchozí analýzy produktů Wget, Teleport Pro, Apache Nutch a Google Mini lze získat určitou představu, co by měl navrhovaný crawler jistě umět, jak by měl crawler vypadat a jak by se mohl ovládat. Tyto vlastnosti do jisté míry souvisí s tím, jak a v jakém prostředí uvažujeme crawler používat. Pro výsledný produkt je dále používanou pracovní jméno WebCrawler. Při návrhu WebCrawleru se tedy pokusím, aby:

- byl kompaktním softwarovým celkem, který bude snadné (nebo alespoň ne zbytečně složité) nainstalovat a používat.
- dodržoval základní ohleduplnost k prohledávanému webu, tj. především nepřetěžovat cílový webový server, popř. vyhnout se extrémním datovým přenosům.
- zvládal základní chování dle požadavků autora cílového webu a neprohledával části webu, kde si to autor nepřeje, popř. bylo rozhodnuto, že to nemá smysl.
- jeho běh rozumně využíval systémových prostředků, především se jedná rozumnou práci s vlákny bez aktivního čekání ve smyčkách.
- umožnil rozložení své práce i mezi vícero počítačů.

5.2 Výběr technologie

Analyzované produkty v této práci ukazují, že webový crawler může mít značně různorodou podobu. Jak již bylo zmíněno, záleží na prostředí a způsobu, který budeme chtít crawler použít...

Jednoduchý crawler pro nějaký specifický problém si můžeme napsat např. v PERLu, Pythoonu, PHP nebo jiném *příjemném* skriptovacím jazyku, kde je podpora pro síťové operace zpravidla dobře ošetřena. K dispozici bývá také propracovaná podpora pro použití regulárních výrazů, což je další nástroj, který pravděpodobně budeme potřebovat. Zajímavá by byla multiplatformnost řešení a poměrně snadno se tak můžeme přiblížit k programu jako Wget.

Realizace programu v podobě Teleportu Pro je také poměrně dobře představitelná. Použil bych např. C++ toolkit Qt, který nabízí možnost vcelku rychle vyrobit elegantní GUI a jeho výborná *výbava* pro programování funkčností, které přímo nesouvisí s GUI (technologie signálů a slotů, procesy a vlákna, podpora pro pohodlnou práci s regulárními výrazy) by vývoj učinila jistě zajímavým. Bezpochyby výborně použitelná by byla i

Java se Swing, popř. .NET. Qt nebo Java by však umožnili opět zachovat jistou multiplatformnost.

V Apache Nutch zajímavě působí přibalená webová aplikace aplikace pro hledání, samotná vyhledávací aplikace (velmi schopná) a crawler jsou však CLI javové programy (JAR archívy), které lze libovolně používat bez aplikačního serveru.

Jak přesně vypadá architektura Google Mini není snadné určit, přesto i tento produkt může posloužit k inspiraci např. způsobem svého ovládání.

Ani jeden z analyzovaných systémů nijak nevyužívá nějaký databázový server. Pro program Wget, popř. Teleport je to pochopitelné, u Nutche by se již dala potřeba nějakého SQL serveru očekávat. Příslušné crawlery si data ukládají na filesystém.

Pro implementaci svého crawleru jsem vybral Javu, která umožní využít rozsáhlých knihoven a různých dalších aplikačních rámců. Použiji databázový SQL server, což umožní snadné ukládání dat a jisté možnosti horizontálního škálování. Data můžou být sdílena mezi více instancemi crawlerů a více crawlerů pak může pracovat společně na jednom úkolu.

5.3 Představa výsledného produktu

Má představa aplikace WebCrawler má podobu J2EE aplikace pro aplikační server Glassfish2. Uvedený AS jsem vybral pro snadnou instalaci a konfigurovatelnost, kterou je možné provádět ve webovém administračním prostředí. Řešení s AS umožní vyvinout lehké webové rozhraní k ovládání, které bude těsně spojeno s robustností a širokými možnostmi práce s Javou. Aplikace může využívat zdrojů a také různá podpůrná řešení v podobě JEE API, které jsou poskytovány nastaveným aplikačním serverem.

Uživatel bude pomocí obyčejného webového prohlížeče přistupovat k webové aplikaci, kde bude moci nastavit projekt, sledovat a řídit základní práci crawleru.

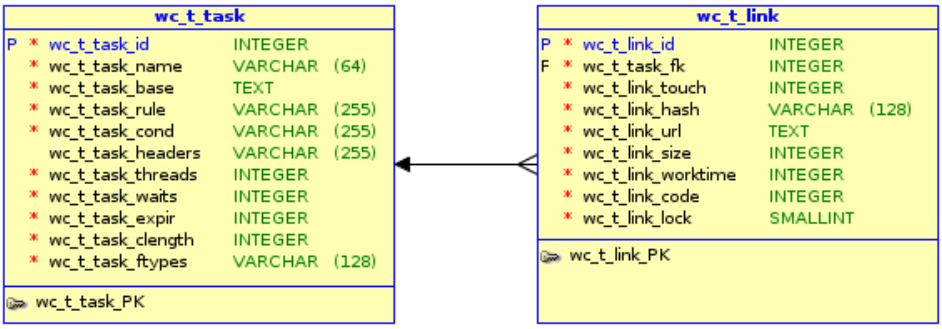
Jako úložiště pro data bude crawleru sloužit databázový server MySQL. Použití AS, který podle J2EE specifikace zpravidla řeší připojení k databázi pomocí JDBC ovladače a datových zdrojů, nás nijak neomezuje jen na MySQL. Jak již bylo zmíněno ve výběru technologie, použití DB umožní škálování celého řešení.

Použití programovacího jazyku Java přirozeně svádí k použití objektově relačního mapování při přístupu k datům v DB. Jako nástroje pro ORM je použito objektového rámce Hibernate. (Datová vrstva aplikace není rozsáhlá, jistě by bylo možné použít i střídmější ORM řešení jako např. iBATIS nebo mít mapování vlastní.)

Je využito aplikačního rámce Spring pro instanciaci a vzájemné provázání některých aplikačních objektů. Dále je použit aplikační rámec Struts2 pro zpracování reakcí na uživatelské akce.

5.4 Postup implementace

Zde jsou představeny (a popř. zdůvodněny) některé důležité části implementace, kterou lze na nejvyšší úrovni rozdělit na návrh datové a aplikační vrstvy.



Obrázek 9: Schéma datové vrstvy aplikace WebCrawler.

5.4.1 Návrh datové vrstvy

Pro datovou vrstvu je použito databázového SQL serveru MySQL. Databázové schéma aplikace (obrázek 9) WebCrawler je vcelku jednoduché. Základem schématu je tabulka pro zadání *procházecí*úlohy (task) pro crawler, který při své práci bude procházet stránky vybraného webu, v nich bude hledat odkazy na další části zkoumaného webu a zmíněné nalezené odkazy budou ukládány do tabulky pro odkazy (link). V tabulce jsou pak jednotlivé odkazy pomocí cizího klíče provázány s vlastní úlohou.

Představme si tabulku pro zadání úlohy, kde jednotlivé atributy představují:

- `id` id úlohy, které je pak cizím klíčem v záznamech odkazů
- `name` jednoduchý název úlohy.
- `base` je první, úvodní (*startovací*) URL úlohy, kde se začne procházet web.
- `rule` pravidlo v podobě regulárního výrazu, které definuje jak rozeznat další odkazy ve stažených datech (zpravidla textových zdrojových souborech) webových stránek. Touto hodnotou tedy popíšeme jak vypadá odkaz na další stránky.
- `cond` další pravidlo ve formě regulárního výrazu, popisující podmínku, kterou musí nalezený (*vyparsovaný*) odkaz splnit, aby byl zahrnut k dalšímu prohledávání. Tato hodnota nám tedy umožní popsat jak má vypadat odkaz, který nás dále zajímá a umožní nám především udržet WebCrawler ve vybrané doméně nebo skupině domén, popř. blíže popsat odkazy, které mají být prohledávány, resp. vyloučit některé typy odkazů.
- `headers` textový řetězec s HTTP hlavičkami, které WebCrawler použije při přístupu k cílovému webovému serveru. Můžeme tak zajistit, že se budou vracet české verze stránek, představit WebCrawler jako nějaký prohlížeč, popř. lze v hlavičce posílat e-mail pro hlášení problémů...
- `threads` hodnotu počtu vláken, která mohou v rámci jedné instace současně pracovat na jedné úloze, tj. počet současně bežících vláken pro zpracování odkazů.

- `waits` hodnotu čekání v ms. před ukončením zpracování (a započetím dálšího stahování) jednoho odkazu. Hodnota je součastí nastavení ohleduplnosti k cílovému webu.
- `expir` čas zadávaný v minutách, za který budou odkazy pod úlohou vyhodnoceny jako zastaralé a budou opět přezkoumány.
- `clength` hodnotu v bajtech, která udává maximální velikost dat jenž cílový webový server nabídne ke stažení, za kterou již obsah nebude stahován. Opět se jedná o jistou ohleduplnost k cílovému webu a současně o ochranu crawleru proti situaci, kdy narazíme např. na odkaz ke stažení 4,7 GB obrazu DVD disku.
- `ftypes` seznam koncovek souborů, se kterými budou případné odkazy *zahozeny*. Nastavení zabrání zbytečnému zpracovávání binárních dat z grafických souborů, souborů flashe a pod.

Následuje význam atributů tabulky pro ukládání odkazů:

- `id` je id odkazu.
- `fk` je cizí klíč svazující nalezený odkaz s úlohou.
- `touch` informuje o času posledního spracování odkazu a podle tohoto atributu se pozná, že odkaz zastaral a je potřeba jej aktualizovat.
- `hash` představuje hodnotu MD5 součtu tvorenou z *id*a *url*, která je v rámci tabulky definována jako unikátní a díky této vlastnosti je zaručeno, že se v DB nebudou v rámci jedné úlohy hromadit stejné odkazy.
- `url` je normalizovaný odkaz nalezený při práci na úloze.
- `size` obsahuje velikost dat stažených v rámci jednoho odkazu, popř. velikost dat nabídnutých cílovým webovým serverem v hlavičce *Content-Length*.
- `worktime` je čas v ms., který trvalo zpracování jednoho odkazu.
- `code` nese informaci o stavu odkazu:
  - `(-1)` kód pro odkaz, který nevyhovuje podmínce pro další zpracování. Získáváme tak informaci o odkazech, které např. vedou mimo cílový web.
  - `(0)` kód pro zatím nezpracovaný odkaz.
  - `(1xx - 5xx)` stavové HTTP kódy zachycené při stahování odkazu.
  - `(600)` kód pro vnitřní chybu stahovače crawleru.
  - `(700, 701)` kód pro odkaz, který nebyl stahován z důvodu překročení maximální velikosti dat pro jeden odkaz podle hlavičky *Content-Length*.
  - `(800, 801)` kód pro odkaz, který je podle pravidla z robots.txt zakázáno procházet

  - `(999)` kód znamenající, že se na odkazu právě pracuje.

- `lock` je hodnota uzamykající odkaz před výběrem ke zpracování. Odkazy jsou zamykány při svém výběru ke zpracování, popř. je možné je možné odkaz zamknout ručně.

5.4.2 Návrh aplikační vrstvy

Aplikační vrstva je realizována jako J2EE aplikace. Návrh aplikační vrstvy je možné přirozeně rozdělit na návrh prezentační vrstvy, která bude představovat pracovní rozhraní pro ovládání crawleru uživatelem, a návrh výkonné vrstvy, která bude představovat a naplňovat aplikační funkčnost. Jistě by bylo možné obě části navrhnout odděleně tak, aby na jednom aplikačním serveru bylo jen ovládací rozhraní prezentační vrstvy pro řízení výkonné vrstvy, umístěné na několika dalších aplikačních serverech. Přestože takové řešení není použito, v aplikaci je mezi prezentační a výkonnou vrstvou zřejmé oddělení do aplikačních balíků.

Aplikační část se skládá ze dvou základních balíků:

- `WebCrawlerEE-war` kde jsou řešeny stránky a akce prezentační vrstvy pomocí aplikačního rámce Struts2. Dále je zde umístěno nastavení pro provázání s výkonnou vrstvou pomocí aplikačního rámce Spring.

- `WebCrawlerEE-ejb` obsahuje třídy výkonné vrstvy, kde je umístěna primární funkčnost a kde se využívá připojení k databázi pomocí rámce pro objektově relační mapování Hibernate.

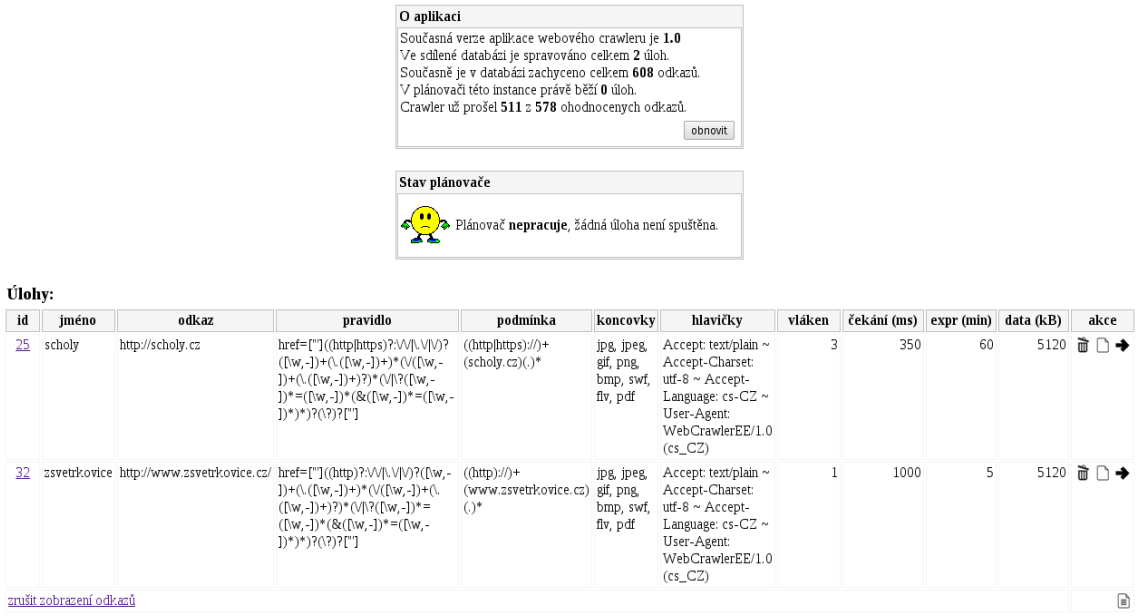
5.4.2.1 Návrh prezentační vrstvy

Pro usnadnění vývoje prezentační vrstvy byl vybrán aplikační rámec Struts2, který je založen na návrhovém vzoru MVC (Model View Controller) a pomáhá v řízení aplikace na základě událostí generovaných uživatelem. Obsahuje taktéž podporu pro validaci webových formulářů, kdy lze pro jednotlivé hodnoty formuláře snadno zadat podmínky a validace se pak stává součástí řízení pomocí událostí.

Uživatel tedy v aplikaci vyvolá akci. Je-li součástí akce od uživatele validace dat, tato se provede. Podle akce (a popř. výsledku validace) se pomocí řídících pravidel vybere příslušná reakce. Se stránkami prezentační vrstvy řešené pomocí Struts2 úzce souvisí tzv. *akční třídy*, kde se právě vykonávají akce spouštěné prací uživatele.

Viditelná část prezentační vrstvy je tvořena dvěma stránkami Struts2 pro ovládání:

- `MainPage.action` na obrázku 10, tvořená pomocí akční třídy `MainPage.java`, s níž je svázána `mainPage.jsp` a slouží k přehledu nad stavem crawleru. Umožňuje přejít k vytvoření nové úlohy a u existujících úloh je možné spustit a zastavit jejich vykonávání. Dále je zde možné úlohu vyčistit a vynutit tak její nové zpracování, popř. úlohu zcela vymazat.



Obrázek 10: Stránka MainPage.action z aplikace WebCrawler se zadanými úlohami.

- `AddTask.action` na obrázku 11, tvořená pomocí akční třídy `AddTask.java`, s níž je navázána `addTask.jsp` a představuje způsob jak v aplikaci vytvořit novou úlohu. Jednotlivé hodnoty formuláře korespondují s atributy výše zmíněné databázové tabulky úloh.

Součástí sady akčních tříd jsou také další třídy pro vymazání, vyčištění, spuštění a zastavení úlohy, ruční uzamykání odkazů, vynulování odkazů a pod. Tyto akce nemají svou vlastní webovou stránku, ale jsou spouštěny z úvodní stránky.

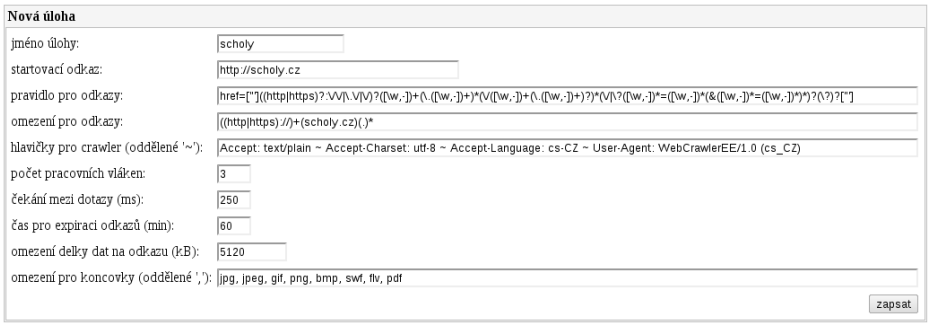
Aplikační balíky z prezentační vrstvy `WebCrawlerEE-war` obsahují:

- `war.view` akční třídy Struts2 pro viditelnou část prezentační vrstvy.
- `war.action` akční třídy pro *neviditelné* akce definované ve Struts2.
- `war.util` některé další pomocné třídy.

Akční třídy frameworku Struts2 představují v prezentační vrstvě aplikace místo, kam jsou získávány informace a odkud jsou *volány* metody tříd z výkonné vrstvy. Takto popsané propojení je realizováno s vyžitím frameworku Spring.

5.4.2.2 Návrh výkonné vrstvy

Návrh výkonné vrstvy zahrnuje třídy a metody, na které je pomocí frameworku Spring navázána prezentační vrstva aplikace. Do těchto tříd a metod jsou dodávány příkazy



Obrázek 11: Stránka AddTask.action z aplikace WebCrawler pro novou úlohu.

uživatele, podle těchto příkazů je pak provedena akce a ovlivněno chování samotného výkonné vrstvy crawleru.

Výhodou použití frameworku Spring je instanciace tříd z výkonné vrstvy aplikace pro prezentační vrstvu a mapovaní instanciovanch objektů do příslušných prezentačních akčních tříd. Spring umožňuje třídy instaciovat pro různé uživatelské úrovně a tak každý uživatel může mít vlastní instanci nějaké třídy, popř. je možné (a v aplikaci používáno), aby uživatelé sdíleli jednu instanci třídy. V případě prostého JSP by bylo nutné zajistit instanciaci tříd ručně, což může být někdy složité a může to vést k nefunkčnosti v podobě běhových výjimek, plýtvání a úniků paměti. Spring třídy instanciuje automaticky sám podle typu v okamžiku kdy je to potřeba a tak usnadňuje a zpřehledňuje vývoj.

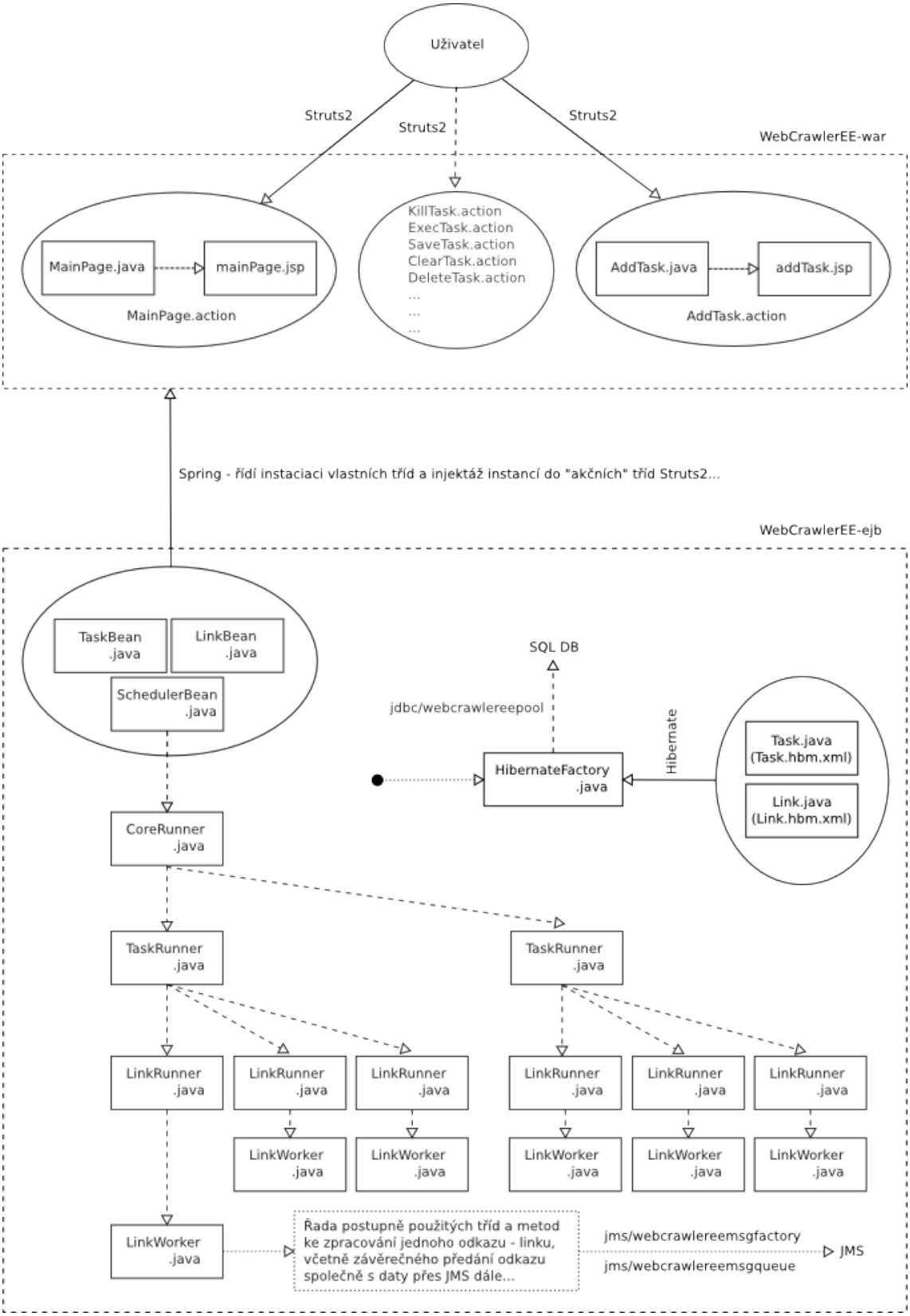
Aplikační balíky z výkonné vrstvy `WebCrawlerEE-ejb` obsahují:

- `ejb.bean` třídy, přes které jsou akce přenášeny do výkonné vrstvy, které jsou instanciovány právě Sprigem a *injektovány* do akčních tříd prezentační vrstvy.
- `ejb.db` třídy objektů a nastavení mapování pro ORM pomocí Hibernate.
- `ejb.runner` třídy pro hlavního správce vláken a plnění úkolů ve vláknech.
- `ejb.util` sadu pomocných tříd, pro logování, stahování dat, parsování dat, hashování, ...

Strukturu aplikace popisuje obrázek č.12, ve kterém je zachycena právě oddělenost prezentační a výkonné vrstvy. Také je zde naznačen způsob práce celé aplikace, jenž je popsán v následující části.

5.4.2.3 Popis práce aplikace

Jistá práce aplikace začíná již spuštěním aplikačního serveru Glassfish, resp. hned po dokončení spouštění dochází k inicializaci nainstalovaných aplikací. V případě aplikace WebCrawler je v aplikačním serveru spuštěn EJB kontejner a s využitím rámce Spring vytvořeny instance tříd `TaskBean` (k práci s objekty úloh), `LinkBean` (k práci s objekty odkazů), `SchedulerBean` (pro přístup k řízení crawleru).



Obrázek 12: Struktura aplikace WebCrawler. - Prezentační a výkonná vrstva.

Spring zajistí vytvoření jediné instance od každé třídy a případný větší počet uživatelů, přistupujících k webovému rozhraní aplikace crawleru, by sdílel právě jednu instanci dané třídy.

Při instanciaci `SchedulerBean` je vytvořena instance třídy `CoreRunner`, která slouží ke správě běžících úloh v rámci jedné (*vlastní*) instance aplikačního serveru a kde dochází k první instanciaci třídy `HibernateFactory`, což vede k vytvoření databázového sezení pomocí rámce Hibernate. Hibernate zajistí mapování databázových tabulek na objekty Task a Link a je nakonfigurován tak, aby využíval tzv. *connection pool* přes zdroj *jdbc/webcrawlereepool* aplikačního serveru, ve kterém je nakonfigurováno samotné připojení k databázi.

Aplikace WebCrawler je teď připravena ke skutečné práci na *procházecích* úlohách a další popis již odráží uspořádání na obrázku struktury aplikace.

Pro uživatele jsou přímo viditelné dvě (již zmíněné) webové stránky, kterými je hlavní stránka s přehledem a stránka s vytvořením nových úloh. Stránky jsou vytvářeny a uživateli poskytovány pomocí aplikačního rámce Struts2. Vlastní stránka je tak vytvořena ze své JSP šablony (část *view*) a akční třídy (část *model*), na kterou je řízení předáváno akcemi (část *controller*) konfigurovanými ve `struts.xml`.

Do jednotlivých akčních tříd, a tak i do samotné prezentační vrstvy, jsou pomocí aplikačního rámce Spring předávány (injektovány) odkazy na instance tříd pro získávání objektů úloh (`TaskBean`), odkazů (`LinkBean`) a objektu pro ovládání výkonné vrstvy crawleru (`SchedulerBean`). V `SchedulerBean` je vedena (a právě řízena) instance třídy `CoreRunner`, která dále nese funkční logiku aplikace.

Hlavní funkčností je provádění procházecí úlohy. To je zajištěno akcí uživatele a předáním ID úlohy skrze akční třídu `ExecTask` do `SchedulerBean`, která ID předá do své instance `CoreRunner`, kde je s pomocí `HibernateFactory` a objektu `Task` provedeno načtení informací o úloze. V `CoreRunner` pak dochází k instanciaci třídy `TaskRunner`, které je předána instance vybrané úlohy `Task`. V `TaskRunner` je vytvářeno vlákno procházecí úlohy a tato se stává spuštěnou.

Při provádění vlákna v `TaskRunner` dochází za sebou k vytváření instancí třídy `LinkRunner`. Každý `LinkRunner` se pokouší z databáze získat odkaz v podobě objektu `Link` (samozřejmě patřící k úloze pod kterou `LinkRunner` spadá) k provedení. Pokud se v `LinkRunner` podaří najít nezpracovaný nebo expirovaný odkaz, vzápětí je vytvořeno vlákno, které zajistí zpracování vybraného odkazu na pozadí.

Nové instance a vlákna `LinkRunner` jsou vytvářeny dokud se daří nalézat nezpracované odkazy a dokud není překročen počet současně běžících vláken pro zpracování odkazů. V pozdější fázi běhu úlohy, kdy už jsou všechny nalezené odkazy zpracovány, se pak hledají expirované odkazy. Nepodaří li se pro `LinkRunner` najít nezpracovaný nebo expirovaný odkaz, dělají se mezi dalším hledám odkazů delší prodlevy.

Uvnitř vlákna `LinkRunner` je vytvořena instance třídy `LinkWorker`, do které je předána instance objektu `Link`. `LinkWorker` pak provádí samotné zpracování odkazu, což bude rozebráno dále. Po ukončení zpracování odkazu je přes `LinkWorker` získáno

několik údajů k aktualizaci záznamu odkazu v databázi a do vlákna `TaskRunner` je oznámeno ukončení jednoho vlákna `LinkRunner`. Vlákno `TaskRunner` se pak pokouší spouštět další vlákna `LinkRunner`.

Cílem byla konstrukce aplikace tak, aby vedle sebe mohlo současně pracovat více úloh a pod každou úlohou pak paralelně daný počet procesů zpracování odkazu. Obrázek struktury aplikace 12 tak představuje stav, kdy pod `CoreRunner` běží 2 úlohy ve vláknech `TaskRunner` a pod každou úlohou běží 3 vlákna `LinkRunner` se zpracováním odkazu. Samotný proces zpracování odkazu probíhá v objektu `LinkWorker`.

5.4.2.4 Popis zpracování odkazu

Samostatnou část tvoří problematika zpracování jednoho odkazu, které začíná předáním objektu `Link` do objektu `LinkWorker`. Výsledkem procesu zpracování odkazu jsou *syrová* stažená data a pak především sada nových odkazů k zařazení do databáze a k dalšímu procházení. Proces zpracování jednoho odkazu probíhá postupně v několika třídách, které řeší různé části, potřeby a problémy. Proces spuštění úlohy a dalšího zpracování odkazů je zachycen diagramem na obrázku 13.

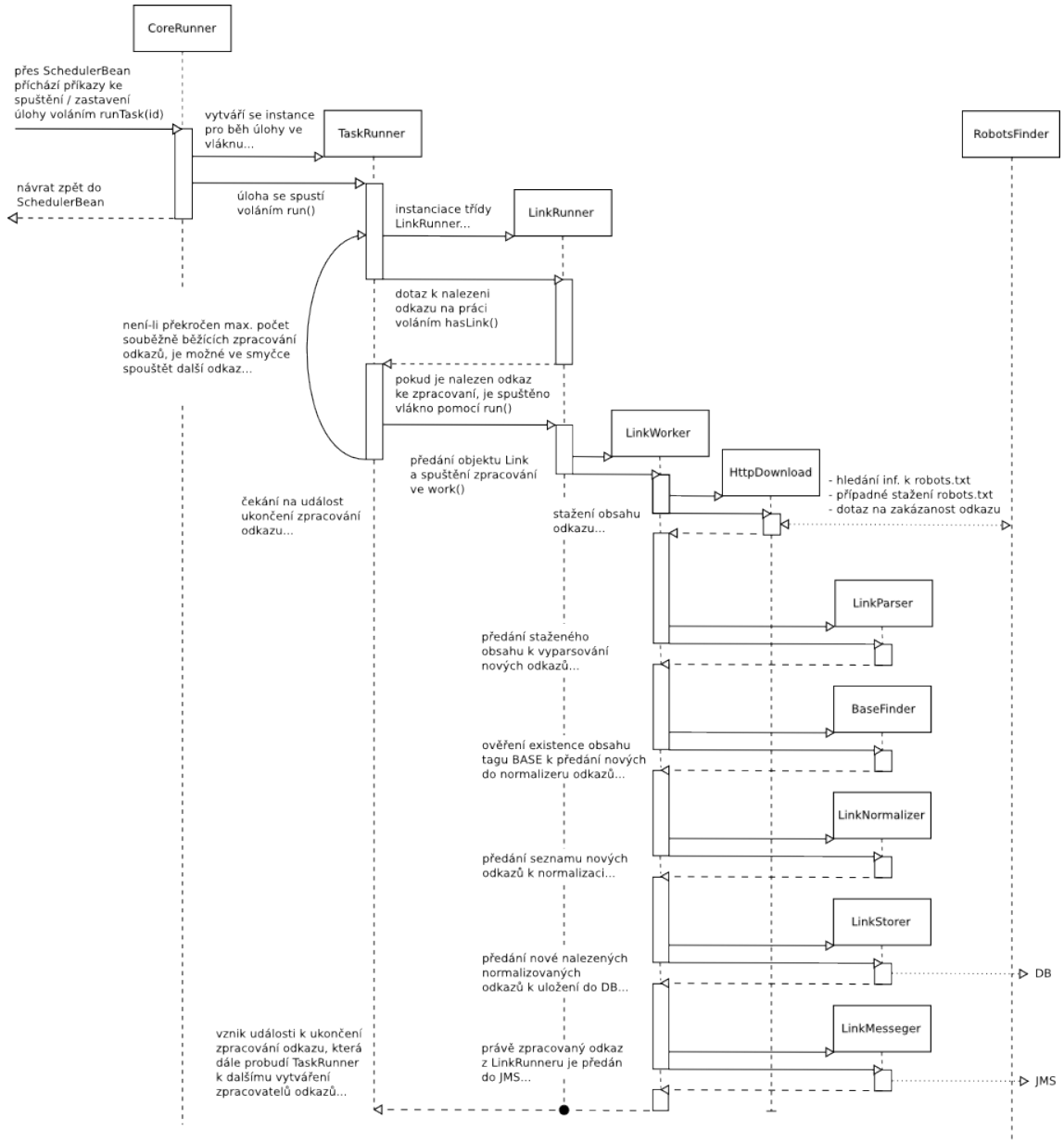
5.4.2.4.1 Stažení dat z odkazu

Zpracování odkazu začíná stáhnutím jeho dat. Stahování dat řeší objekt `HttpDownloader`. Konstruktor očekává URL a výsledkem práce jsou stažená data. Pokud se data nepodaří stáhnout, nepokračuje se v dalším zpracování odkazu a do databázového záznamu k odkazu je zapsán chybový kód. Stahovač podporuje pouze HTTP. Do stahovače je možné nahrát také hlavičky, kterými se představí cílovému webovému serveru.

Přesto, že národní znaky byly pro českou doménu výrazně zamítnuty [4], je možné se touto vlastností na internetu setkat. WebCrawler umožňuje stahování z domén obsahující v názvu znaky národních abeced. Řetězec s národní URL je pak přeložen z UTF do mezinárodní URL v ACE [5] (ASCII compatible encoding), což je popsáno odpovídající normou. URL je tak připravena ke stahování.

Ještě před stahováním se hledá, zda vybraná URL není zakázána podle `robots.txt`. Je-li zakázána, stahování se přeruší a k odkazu je opět doplněn zvláštní kód. Kontrola probíhá v objektu `RobotsFinder`. Zde se z kontrolované URL vybere doména (crawler totiž může přecházet také mezi doménami a jejich různými úrovněmi), kde se pak zkusí najít soubor `robots.txt`. `RobotsFinder` si pamatuje kde zkoušel soubor `robots.txt` hledat a pokud jej předtím nenalez, nezkouší jej hledat znovu, resp. se o to pokusí až za dlouhou dobu, aby zbytečně neobtěžoval cílový webový server. Soubory robots.txt mohou nést různá omezení pro různé crawlery [2], často však je však použito zobecnění ve formě * a WebCrawler proto koumá jen řádky s *Disallow:*.

Je sestaveno spojení na URL a získají se HTTP hlavičky. Zde je pomocí hlavičky `Content-Length` ověřeno, nepřesáhne-li velikost předaných dat maximální velikost stahovaných dat na odkaz definovanou v zadání úlohy. V případě překročení hodnoty



Obrázek 13: Sekvenční diagram spuštění úlohy a další proces zpracování odkazů.

se opět vytvoří záznam s chybovým kódem, jinak jsou data z odkazu stažena a předána zpět do objektu `LinkWorker`.

5.4.2.4.2 Extrakce (parsování) nových odkazů ze stažených dat

Další fází zpracování odkazu je extrakce nových odkazů ze stažených dat. Data jsou předána do objektu `LinkParser`. Ten je vytvořen se zadáním a zkompilováním pravidla ve formě regulárního výrazu pro rozeznání odkazů. Pravidlo je součástí zadání nové úlohy a kompilací je vytvořena instance třídy `Pattern` z knihoven Java API.

Instance třídy `Pattern` je dále použita při vytváření instance třídy `Matcher` (opět součást knihoven Java API), která je ve smyčce volána pro získání jednotlivých řetězců (nových odkazů) podle *regexpu* ze stažených dat. Nové odkazy jsou ukládány do objektu `ArrayList`, který je pak vrácen do `LinkWorker`.

Zvolený způsob parsování odkazů se zdá poměrně jednoduchý. Schopnost rozeznat odkazy není zabudována do aplikace a je možné ji různě ovlivnit v nastavení úlohy. Na druhou stranu je potřeba jistá šikovnost uživatele při sestavení odkazu. Formulář pro zadávání nové úlohy obsahuje přednastavený regulární výraz, který je možné rozšiřovat. Vhodným rozšířením může být např. doplnění řetězce `|action` za `href`, což do nalezených odkazů zahrne také odkazy z webových formulářů tvořených tagem FORM..

Zde je však potřeba také upozornit na nutnost sestavení regulárního výrazu rozpoznávajícího odkazy tak, že samotný nový odkaz vedoucí k dalšímu dokumentu je součástí nějakého parametru HTML tagu, typicky parametr *href*, *action*a pod... Např. volně napsané *http://www.vsb.cz* v textu stránky mimo jakýkoliv tag nebude rozeznáno. Předpoklad existence zápisu odkazu jako parametru v nějakém tagu souvisí částí parsovacího procesu, kdy z takto získaných odkazů v parametru je extrahován samotný odkaz..

Parsování odkazů je jednou z důležitých částí aplikace crawleru a je zde nutno si poměrně dobře rozmyslet co přesně chceme. Domnívám se, že dosažení naprosté univerzálnosti v jednom parseru odkazů může být problematické, pak je jistě možné proces a způsob parsování dále vylepšit implementací vícero specializovaných parserů a jejich zřetězením za sebe.

5.4.2.4.3 Normalizace nových odkazů

Pokračováním procesu zpracování odkazu je tzv. normalizace nalezených odkazů. Jde o skutečnost, že rozeznané a nasbírané odkazy z objektu `LinkParser` , jsou všechny v původním tvaru tak, jak byly vyčteny ze stažených dat. Znamená to že odkazy mají tvar např. *http://www.vsb.cz/*, *www.fei.vsb.cz/dokument.html*, ale také mohou být ve tvaru *./dokument.html* nebo *../../dokument.html*.

První odkaz *ukazuje* cestu ke svému dokumentu velmi přesně. U druhého odkazu chybí popis protokolu, ale není problémem předpokládat, že se jedná o *http*. Tyto odkazy jsou *absolutními* odkazy a plně popisují cestu k dokumentu, tj. obsahují protokol, server a popř. port, dokument a dále možné parametry. Jistým problémem pro zaznamenání od-

kazu do databáze crawleru jsou *relativní*odkazy, které jsou prezentovány druhými dvěma ukázkami. Tyto odkazy je potřeba normalizovat, tj. doplnit o plnou cestu k cílovému dokumentu, neboť prostým uložením nenormalizovaného odkazu do databáze crawleru (alespoň v jeho současné implementaci) se tato informaci jistě ztratí.

Normalizace odkazů probíhá v objektu `LinkNormalizer`, kam se předává `ArrayList` nových odkazů a následně je možné získat `ArrayList` normalizovaných odkazů.

Ještě před samotnou normalizací sady nových odkazů v `LinkNormalizer`, jsou stažena data dokumentu prohledána objektem `BaseFinder`, který v hlavičce webové stránky hledá tag BASE. Tento tag slouží k doplnění všech relativních odkazů na stránce tak, aby se z nich staly právě odkazy absolutní. Mohlo by se zdát, že tag řeší problém s normalizací relativních odkazů, nicméně jeho zápis ve stránce není povinný, a tak chybí-li, je potřeba normalizaci zvládnou jiným způsobem. Je-li tento tag nalezen, pak jeho hodnota je předána k normalizaci relativních odkazů.

Při normalizaci nového odkazu se vychází ze znalosti *zdrojového* odkazu (samozřejmě již normalizovaného) dokumentu, na kterém byl nový odkaz nalezen, resp. ze kterého se na nový odkaz *přišlo*. Při normalizaci odkazu se postupuje v několika krocích...

- Pokud normalizovaný odkaz začíná na `./` a je-li k známa hodnota z tagu BASE, pak se odkaz doplní o řetězec z BASE.
- Řetězec odkazu se dále použije pro vytvoření objektu URL z knihoven Java API.
  - Podaří-li se vytvořit objekt URL, zavolají se metody `toURI()` a `normalize()`.
  - Dále se odkazu odstraní případné poslední lomítko.
- Jestliže dojde při vytváření objektu URL k výjimce, znamená to, že řetězec s odkazem nesplňuje vlastnosti použitelného odkazu a objekt URL z něj nelze vytvořit.
  - Pak je k odkazu, který je právě normalizován, připojena URL zdrojového dokumentu oříznutá o část popisující soubor dokumentu.
  - Upravený odkaz je opět použit ke konstrukci objektu URL.

Volání metody `normalize()` objektu URI z Java API zajistí, že:

- Všechny segmenty `.` jsou odstraněny.
- Všechny segmenty `..` jsou odstraněny společně s jim předcházející částí v části URL, která cestou k souboru na serveru. [3] To je opakováno dokud je to možné, tj. dokud je před segmenty část, kterou s nimi lze *svázat* a společně je odstranit.

Normalizovaná URL by neměla obsahovat žádné `.` nebo `..` segmenty. Konečným výsledkem procesu normalizace v rámci objektu `LinkWorker` je sada nově nalezených normalizovaných odkazů v objektu `ArrayList`.

5.4.2.4.4 Uložení nových odkazů

Proces zpracování odkazu pokračuje uložením sady nově nalezených odkazů do databáze. Ukládání je prováděno v objektu `LinkStorer`, kam je předána sada odkazů v `ArrayList`.

Odkaz k uložení je porovnán se sadou zakázaných koncovek, které jsou definovány při zadávání úlohy. Pokud je koncovka odkazu zakázaná, odkaz je zahozen.

Pro každý nový odkaz je vytvořen objekt `Link` a pro URL je spočítán MD5 hash. Následně je odkaz porovnán s regulárním výrazem *omezení pro odkazy*, který je zadáván při vytvoření úlohy, a který popisuje dále průchozí (povolené) URL. Jestliže URL vyhoví jako dále průchozí, je takto označena, jinak objekt odkazu zůstane se značkou jako dále neprůchozí. Důvodem pro ukládání i dále neprůchozích URL do databáze je prostá snaha o přehled i nad těmito URL.

Novému objektu `Link` jsou nastaveny některé výchozí vlastnosti, tj. nulová velikost dat, kód nezpracování 0 (nebo že nebude zpracováván -1), zámek a `Link` je uložen.

Při ukládání nového odkazu do příslušné tabulky může dojít k výjimce jedinečnosti atributu *wc_t_link_hash*, což je situace, která je očekávána a tímto způsobem je vyloučeno ukládání duplicitních odkazů do databáze.

Odkazům jsou ponechávány *parametrické* části URL. Tato věc by za jistých okolností mohla být problémem, pokud by se crawler zacyklil na nějakém zvláště dynamicky generovaném odkazu, např. obsahujícím časové razítko, kdy by takový odkaz by pro crawler stále nový.

V databázi je pro záznamy s odkazy používáno přírůstkového vytváření ID přímo databázovým serverem, čímž je umožněna jednoduchá práce více crawlerů na jedné úloze, protože nevzniknou záznamy se stejným ID. Načítání stejných odkazů ke zpracování více než jedním crawlerem je omezeno zamykacím atributem *wc_t_link_lock* a možné uložení stejných nových odkazů ze dvou crawlerů je omezeno již zmíněným způsobem s hashováním URL.

5.4.2.4.5 Další zpracování odkazu a jeho dat, volání JMS

V úvodu práce, byl webový crawler popsán jako systém k procházení struktury dokumentů, kde jsou vyhledávány odkazy na další dokumenty, aby i tyto mohly být dále prohledány na odkazy k dalším dokumentům... Samo o sobě však takové procházení nemá žádný hlubší smysl.

Aplikace WebCrawler proto obsahuje implementaci použití jednoho z mechanizmů platformy Java EE, pomocí kterého je URL s daty dokumentu předána k dalšímu zpracování. Uvedeným mechanizmem je JMS (Java Messaging Services), jenž umožňuje posílání asynchronních zpráv mezi komponentami, resp. celými aplikacemi uvnitř aplikačního serveru. K tomuto účelu jsou v aplikačním serveru nakonfigurovány dva zdroje *jdbc/webcrawlereemsgfactory* a *jdbc/webcrawlereemsgqueue*, skrze které je komunikace mezi

aplikacemi prováděna. Výhodou je jistá jednoduchost použití.

Zásadním smyslem tohoto řešení je však snaha o oddělení implementace webového crawleru a jakéhokoliv dalšího zpracování dat. Jako jednoduchého referenčního příkladu je zde použito vytvořené aplikace WebPark, která v aplikačním serveru vedle WebCrawleru čeká na zprávy ze zdrojů. Z obdržených zpráv přebírá URL a data, která si uloží do vlastní databáze, nad kterou je pak umožněno velice jednoduché vyhledávání pomocí dotazů SELECT a LIKE.

Posledním krokem ve zpracování odkazu v objektu `LinkWorker` je tedy odeslání zprávy do fronty aplikačního serveru. `LinkWorker` je ukončen a řízení je předáno zpět do objektu `LinkRunner` , kde se před ukončením vlákna počká jistou dobu *čas čekaní mezi dotazy*, definovanou při vytváření úlohy. `LinkRunner` je ukončen a `TaskRunner` může nechat zahájit zpracování dalšího odkazu v novém objektu `LinkRunner`.

6 Instalace aplikace WebCrawler

Instalaci aplikace WebCrawler lze svou složitostí přirovnat k instalaci aplikace Apache Nutch, která je v této práci uvedena. Při popisu postupu budou prováděny také kroky potřebné k instalaci aplikace doplňkové aplikace WebPark, která je svou konstrukcí podobná WebCrawleru.

6.1 Instalace databáze MySQL

Jako databáze použita MySQL. Instalace je jednoduchá, lze si vystačit s výchozím instalačním balíčkem, který je dostupný na http://www.mysql.com/downloads/mysql/. Server nainstalujeme, spustíme a ověříme, že se na něj připojíme. K práci s MySQL lze doporučit např. grafický nástroj *MySQL Administrator* a *MySQL Query Browser*.

Přílohou k této bakalářské práci jsou dva krátké SQL skripty `WebCrawlerDB.sql` a `WebParkDB.sql` , které vytvoří uživatelské účty s hesly, databázové schémata a také účtům přidělí potřebná práva na tato schémata.

Připojíme se k databází pomocí nástroje *MySQL Query Browser* a necháme provést skript `WebCrawlerDB.sql`, popř. je možné skript spustit z příkazové řádky pomocí `mysql -user=root < WebCrawlerDB.sql` (na platformě Windows bývá heslo roota prázdné). Podobně postupujeme také pro skript `WebParkDB.sql`.

Existující databázová schémata a uživatelské účty jsou potřebné pro další bezproblémové vytvoření *connection poolů* na aplikačním serveru, proto ověříme, že se k databázi můžeme připojit s uživatelem *webcrawler* s heslem *webcrawler* a uživatelem *webpark* s heslem *webpark*.

6.2 Instalace aplikačního serveru Glassfish2

K instalaci aplikačního serveru Glassfish2 je potřeba funkční JDK, testováno na verzi 1.6.0 update 21. Verze aplikačního serveru je *Sun GlassFish Enterprise Server v2.1.1 ((v2.1 Patch06)(9.1_02 Patch12)) (build b31g-fcs)*. Potřebný instalační balíček je dostupný na https://glassfish.dev.java.net/downloads/install/v2.html.

`$ java -Xmx256m -jar filename.jar` (rozbalíme instalační balíček)
`$ cd glassfish` (vstoupíme do adresáře s rozbalenou instalací aplikačního serveru)
`$ chmod -R+x lib/ant/bin` (na Linuxu nebo Unixu přidělíme spouštěcí právo...)
`$ lib/ant/bin/ant -f setup.xml` (...a necháme naistalovat Glassfish)
`$ lib\ant\bin\ant -f setup.xml` (na Windows jen nainstalujeme)

Glassfish spustíme pomocí programu `asadmin` v adresáři `/opt/glassfish/bin`:

```
$ asadmin start-domain domain1
```

V `/opt/glassfish/domains/domain1/logs/server.log` můžeme sledovat log aplikačního serveru. Sledování logu je velmi užitečné, neboť také sama aplikace WebCrawler svou práci logováním bohatě popisuje. Na konci instalace ověříme dostupnost administračního rozhraní na http://localhost:4848.

6.3 Konfigurace aplikačního serveru Glassfish2

Před nahráním aplikací je potřeba aplikační server nakonfigurovat. Jedná se především o vytvoření dvou *connection poolů* pro aplikace WebCrawler a WebPark (připomeňme, že WebPark má vlastní databázi) a vytvoření JMS zdrojů/front pro předávání zpráv z WebCrawleru do WebParku. Webovýcm prohlížečem se přihlásíme do administračního rozhraní na http://localhost:4848, výchozí jméno je *admin* a heslo *adminadmin*.

Začneme konfigurací databázových connection poolů: V postranní nabídce postupně vybereme *Resources* → *JDBC* → *Connection Pools* → tlačítko *New...* **Name** pro pool volíme např. *WebCrawlerDataPool*, **Resource Type** vybere *javax.sql.ConnectionPoolDataSource*, **Database Vendor** vybereme *MySQL*. Klikneme na *Next* a poté vytvoříme nový connection pool uložením přes *Finish*.

Connection pool je potřeba ještě ručně nastavit. V seznamu connection poolů vybereme právě vytvořený *WebCrawlerDataPool* a na záložce *Additional Properties* přidáme vlastnosti **User** = *webcrawler*, **Password** = *webcrawler* a hodnoty **URL** a **Url** nastavíme na *jdbc:mysql://localhost:3306/WEBCRAWLER?useUnicode=true&characterEncoding=UTF-8*

Stejným způsobem nastavíme také druhý connection pool pro aplikaci WebPark, jen na odpovídajících místech místo *WebCrawler* a *crawler* volíme řetězce *WebPark* , resp. *park*. Taktéž heslo do databáze pro WebPark je *webpark*.

Funkčnost connection poolů ověříme v záložce *General* pomocí tlačítka *Ping* a k funkčním connection poolům dále nastavíme odpovídají JDBC zdroje.

V nabídce *Resources* → *JDBC* → *JDBC Resources* → tlačítko *New...* **JNDI Name** pro aplikaci WebCrawler musí být přesně *jdbc/webcrawlereepool*, **Pool Name** vybere *WebCrawlerDataPool*, který jsme vytvořili v odstavci výše. Můžeme si nastavit nějaký **Description** a pro **Status** zatrhneme *Enabled*. Obdobným způsobem nastavíme také JDBC zdroj pro aplikaci WebPark... Názvy JDBC zdrojů se používají v aplikacích a tedy je potřeba, aby byly při instalaci samotných aplikací dostupné.

Pokračujeme konfigurací JMS front: 1) V nabídce *Resources* → *JMS Resources* → *Connection Factories* → tlačítko *New...* **JNDI Name** nastavíme *jms/webcrawlereemsgfactory*, **Resource Type** vybereme *javax.jms.TopicConnectionFactory*, můžeme si nastavit **Description** a dáme **Status** jako *Enabled*. 2) V nabídce *Resources* → *JMS Resources* → *Destination Resources* → tlačítko *New...* **JNDI Name** nastavíme *jms/webcrawlereemsgqueue*, **Physical Destination Name** vyplníme *WebCrawlerMsgQueue*, **Resource Type** vybereme *javax.jms.Topic*, ještě **Description** a **Status** na *Enabled*.

Pro lepší práci s aplikačním serverem doporučuji ještě v nabídce *Application Server*, záložka *JVM Settings*, podzáložka *JVM Options* nastavit parametr *XX:MaxPermSize=192m* na hodnotu 386 nebo 512. Vyhneme se tak vyčerpání paměti *PermGen* v rámci Java JVM.

Máme-li nainstalovanou databázi a nakonfigurovaný aplikační server s fungujícími connection pooly, můžeme přikročit k instalaci (přesněji *deploy*) aplikace WebCrawler a WebPark do aplikačního serveru.

6.4 Instalace aplikace WebCrawler (a WebPark) do AS Glassfish2

V rámci příloh u této bakalářské práce jsou na CD připraveny dva aplikační EAR archívy, WebCrawlerEE.ear a WebParkEE.ear.

Nejprve do `/opt/glassfish/domains/domain1/autodeploy` nakopírujeme aplikační archív WebCrawlerEE.ear a sledujeme log aplikačního serveru, který je umístěn v `/opt/glassfish/domains/domain1/logs/server.log`. Glassfish zaznamená nový instalační archív, ten rozbalí do struktury serveru a tak provede tzv. deploy aplikace...

```
[#|2011-04-26T17:25:41.574+0200|INFO|sun-appserver2.1|Log|_Thread
ID=19;_ThreadName=Timer-10;|ejb.bean.SchedulerBean@122f665: Halo!|#]
[#|2011-04-26T17:25:41.575+0200|INFO|sun-appserver2.1|Log|_Thread
ID=19;_ThreadName=Timer-10;|ejb.runner.CoreRunner@1941421: Halo!|#]
[#|2011-04-26T17:25:42.198+0200|INFO|sun-appserver2.1|Log|_Thread
ID=19;_ThreadName=Timer-10;|ejb.bean.TaskBean@52ae21: Halo!|#]
[#|2011-04-26T17:25:42.200+0200|INFO|sun-appserver2.1|Log|_Thread
ID=19;_ThreadName=Timer-10;|ejb.bean.LinkBean@e60568: Halo!|#]
[#|2011-04-26T17:25:42.653+0200|INFO|sun-appserver2.1|javax.enterprise
.system.tools.deployment|_ThreadID=19;_ThreadName=Timer-10;|[AutoDeploy]
Successfully autodeployed : /opt/glassfish2/domains/domain1/autodeploy/
WebCrawlerEE.ear.|#]
```

Můžeme zde vidět instanciaci tříd, jak je popisována v kapitole 5.4.2.3 a současně závěr výpisu obsahuje hlášku aplikačního serveru o úspěšném deployi aplikace. Dále stejným způsobem nainstalujeme aplikaci WebPark. Opět nakopírujeme aplikační archív WebParkEE.ear do adresáře `autodeploy`, přičemž v logu můžeme sledovat:

```
[#|2011-04-26T17:45:39.009+0200|INFO|sun-appserver2.1|Log|_Thread
ID=19;_ThreadName=Timer-10;|ejb.bean.SearchBean@1becc3c: Halo!|#]
[#|2011-04-26T17:45:39.030+0200|INFO|sun-appserver2.1|Log|_Thread
ID=19;_ThreadName=Timer-10;|ejb.bean.MessageBean@17b0401: Halo!|#]
[#|2011-04-26T17:45:39.429+0200|INFO|sun-appserver2.1|javax.enterprise
.system.tools.deployment|_ThreadID=19;_ThreadName=Timer-10;|[AutoDeploy]
Successfully autodeployed : /opt/glassfish2/domains/domain1/autodeploy/
WebParkEE.ear.|#]
```

Výpis obsahuje opět oznámení úspěšného deploye aplikace, kterému předchází dvě aplikační hlášení *Halo!* o instanciaci. [4] První hlášení je z objektu `SearchBean`, který se v aplikaci WebPark stará o přijetí požadavků na vyhledávání z prezentační vrstvy a vrácení výsledků. Druhé hlášení je z objektu `MessageBean`, kde je implementována funkčnost vyčkávání na zprávy. Přijatá URL a data ze zpráv, jsou následně uložena pomocí mapování přes framework Hibernate do tabulky s atributy *wp_t_page_hash* (jedná se opět o MD5 součty z dodané URL), *wp_t_page_url* a *wp_t_page_data*.

Tímto je instalace WebCrawleru a testovací aplikace WebPark dokončena. Aplikační server Glassfish může v průběhu nastavování vyžadovat restart. Obě aplikace jsou v aplikačním serveru vidět v nabídce *Applications* → *Enterprise Applications*.

Aplikace WebCrawler svou práci poměrně bohatě popisuje v logu aplikačního serveru. Je možné sledovat instanciaci důležitých objektů a průběh důležitých procesů při práci, kde patří především výběr odkazů ke zpracování, stahování dat, normalizace odkazů, zasílání zpráv přes JMS. Stejně tak aplikace WebPark oznamuje některé části své práce, opět instanciaci objektů a především zachycování zpráv.

[4] Metoda jednoduchých zvolání v konstruktoru se mi osvědčila při seznamování se různými frameworky. Často mě zajímá kdy které objekty vznikají, což pomáhá udržet přehled nad průběhem akcí v aplikaci. Společně s hlášením jsou užitečné i výpisy ukazatelů na objekty, např. *ejb.bean.MessageBean@17b0401*.

7 Použití aplikace WebCrawler

V následující části bude představena aplikace WebCrawler. Ukážeme si přístup k aplikaci, uživatelské rozhraní, založení nové úlohy a možnosti ovládání. Nakonec si bude představena testovací aplikace WebPark.

7.1 Hlavní stránka aplikace WebCrawler

Aplikace je přístupná přes webový prohlížeč na (pokud byla instalována na osobním počítači) http://localhost:8080/WebCrawlerEE-war/, kde proběhne přesměrování na hlavní stránku aplikace http://localhost:8080/WebCrawlerEE-war/pages/MainPage.action. Na obrázku 14 je ukázka úvodní stránky aplikace.

V rámečku *O aplikaci* se lze dozvědět několik základních informací o využití aplikace a postupně to jsou:

- Současná verze aplikace webového crawleru.
- Počet úloh, které jsou založeny ve společné databázi.
- Počet všech odkazů, které crawler nalez na zadaných stránkách a které jsou zařazeny do databáze. Zde spadají také odkazy, které nebudou dále procházeny, ale nad kterými (alespoň jsem se takto při návrhu rozhodnul) chci mít přehled.
- Počet právě pracujících úloh. V crawleru je možné mít nadefinováno několik úloh, které můžeme dle potřeby spouštět nebo mít zastavené.
- Počet zpracovaných odkazů z odkazů, které budou zpracovány. Z počtu odkazů, které budou zpracovány, jsou vyjmuty odkazy, které jsou označeny jako dále neprůchozí podle pravidla *omezení pro odkazy*, zadávaného při zakládání nové úlohy.

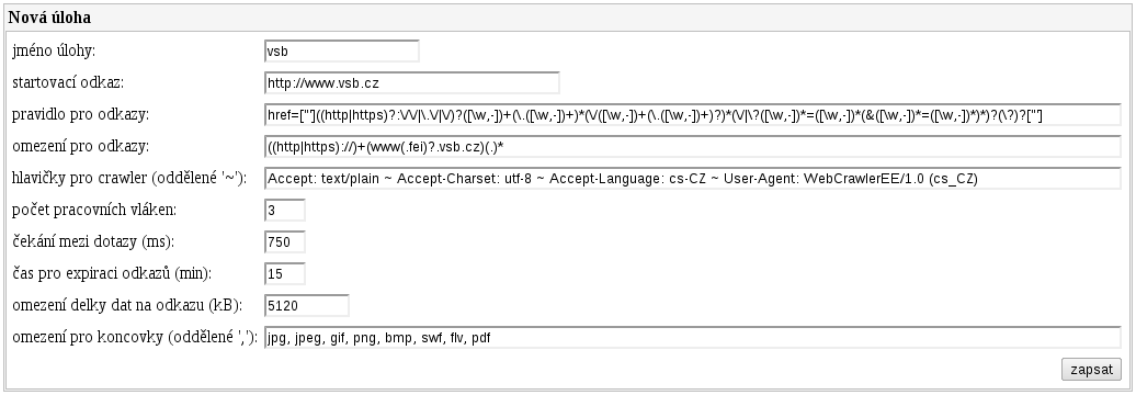
Další rámeček *Stav plánovače* je jednoduchým indikátorem aktivity crawleru. Případný redeploy aplikací v aplikačním serveru by neměl být prováděn při spuštěných úlohach. Je-li alespoň jedna úloha spuštěna, tj. obsahuje-li `CoreScheduler` alespoň jeden objekt `TaskRunner` s běžícím vláknem, pak panáček skáče.

Poslední částí hlavní stránky je tabulka s přehledem založených úloh k procházení. V oblasti tabulkového atributu *akce* (vpravo dole) je ikona k založení nové úlohy.

7.2 Založení nové úlohy v aplikaci WebCrawler

Na stránce http://localhost:8080/WebCrawlerEE-war/pages/AddTask.action, kam se dostaneme kliknutím na ikonu *přidat* vpravo dole v tabulce úloh, můžeme založit novou úlohu. Projděme si jednotlivé položky vytvářecího formuláře, společně s problematikou jejich nastavení, na příkladu vytvoření nové úlohy k procházení stránek VŠB-TUO podle obrázku 15:

Obrázek 14: Úvodní stránka aplikace WebCrawler, ještě bez úloh.



Obrázek 15: Formulář s vytvářenou úlohou pro stránky VŠB-TUO.

- **jméno úlohy** slouží k prostému pojmenování úlohy. Jméno by mělo obsahovat jen znaky A-Z, a-z, 0-9. Vyberme např. *vsb*.
- **startovací odkaz** je odkazem, na kterém se začne v úloze procházet cílový web. Hodnota pole musí splňovat tvar URL adresy. Za normálních okolností volíme úvodní stránku webu a v našem příkladu to bude *http://www.vsb.cz*.
- **pravidlo pro odkazy** je regulární výraz popisující řetězce, které v kódu webových stránek nesou informaci o URL adresách na další dokumenty. Běžně to jsou parametry HREF tagu A (anchor), tj. řetězce jako *href="..."*. Odkazy na další stránky však mohou být také obsahem parametru ACTION z tagu webových formulářů FORM, popř. parametry HREF jsou často také parametry tagu LINK s odkazy na soubory s CSS styly v HTML hlavičce stránky... WebCrawler nerozlišuje, ve které části stránky (HEAD, BODY, FORM) odkaz našel.

  Formulář obsahuje předvyplněný REGEXP, který se v průběhu testování aplikace WebCrawler osvědčil, a na kterém již není potřeba provádět zvláštní úpravy. Příkladem vhodné úpravy může být např. doplnění řetězce *href=...* na řetězec *href|action=...* , což umožní získání URL právě z formulářových tagů.
- **omezení pro odkazy** je opět REGEXP popisující URL, které pokud mu vyhoví, budou povoleny k dalšímu procházení. Nastavení omezení je důležité pro *udržení* WebCrawleru v mezích jistého prostoru, kterým je zpravidla doména. Pole formuláře opět obsahuje přednastavený REGEXP, který se ukázal (většinou) jako vyhovující. V uvedeném výrazu je potřeba nahradit část řetězce *DOMAIN* za doménu, kterou chceme prohledávat, tj *DOMAIN* přepíšeme na *www.vsb.cz*.

  Ovšem pozor! Typicky rozsáhlejší webové stránky (ke kterým se web VŠB-TUO jistě dá zařadit) mohou mít více domén. Dostáváme se tak do situace, kdy odkazy na *www.fei.vsb.cz* **nevyhoví** uvedené podmínce a nebudou dále prohledávány. Upravit REGEXP, abychom mohli procházet web VŠB-TUO včetně domény FEI, můžeme např. tak, že řetězec *DOMAIN* nahradíme *www.*.vsb.cz*.

  Nyní bude WebCrawler již procházet také doménu FEI. Ovšem opět pozor! K procházení bude zahrnuta také doména HGF (*www.hgf.vsb.cz*), což nám nemusí vyhovovat. Je nutné upravit REGEXP např. náhradou *DOMAIN* za *www(|.fei).vsb.cz* nebo *www(.fei)?.vsb.cz*. Tak bude procházeno vše v doméně *www.vsb.cz* a *www.fei.vsb.cz*. [5]
- **hlavičky pro crawler** obsahují sadu HTTP hlaviček, kterými se WebCrawler identifikuje na cílovém webovém serveru. Hodnota nevyžaduje zvláštní úpravy.
- **počet pracovních vláken** představuje hodnotu pro maximální počet současně běžících zpracování odkazu, tj. počet vláken objektů `LinkRunner`. Výchozí hodnota je 1, běžné je použití hodnoty asi 5, nastavme 3.

[5] Práce s regulárními výrazy vyžaduje jistou obratnost. Pro testování je dobré použít nějaký validátor regulární výrazů, na kterém si můžeme ověřit funkčnost námi navrženého výrazu v (i rozsáhlejším) textu. K tomuto účelu se mi velice osvědčil elegantní validátor v javascriptu na http://regexpal.com/.

Obrázek 16: Nová úloha na hlavní stránce aplikace WebCrawler.

- **čekání mezi dotazy** je čas v milisekundách, po který se počká (vlákno se uspí) než je dokončena úloha zpracování a započne se volání nového zpracování. Výchozí hodnota je 1000, ale vhodné jsou i kratší hodnoty od 250. My nastavíme 750 ms.
- **čas pro expiraci dotazů** je doba, za kterou bude již prohledaný odkaz znovu vybrán k prohledání. Každý odkaz v databázi má časové razítko svého poslední zpracování. Pokud již WebCrawler nenalézá nezpracované odkazy, začne ve své databázi hledat expirované (avšak nezakázané) dotazy k opětovnému zpracování. (Vyhledání nových odkazů.) Výchozí hodnota je 60, my nastavíme 15 minut.
- **omezení delky dat na odkazu** je hodnota v kB, která omezí stahování dat nad velikostí, kterou oznámí cílový webový server v HTTP odpovědi v `Content-Length`. Ponechejme výchozí hodnotu 5120 kB, tj. 5 MB.
- **omezení pro koncovky** je poslední ze zadávaných parametrů k vytvoření úlohy, sloužící k vyloučení odkazů na obrázky, binární data a nebo data, o kterých víme, že to nemá smysl, popř. si s nimi WebCrawler *neporadí* a nebude v nich schopen hledat další odkazy. (Mohla by však být vyžadována funkčnost, která zajistí předání těchto dat k dalšímu zpracování.)

Kliknutím na *Zapsat* vytvoříme novou úlohu, která se ihned zobrazí na úvodní stránce podle obrázku 16. V databázi odkazů úlohy je automaticky založen první/startovací odkaz.

7.3 Spuštění a přehled nad stavem jednotlivých úloh

V tabulce úloh nyní vidíme novou úlohu *vsb*. Informace *O aplikaci* uvádějí jednu úlohu v crawleru a dále je v databázi jeden (právě onen *startovací*) odkaz, který je nezpracovaný, tj. 0 nezpracovaných odkazů z celkového množství 1 odkazů v databázi odkazů. Cralwer

23	1	2011-04-27 16:57:24.004	2adc3ec4794c13d96f345512d196633	http://www.vsb.cz/cs/okruhy/kontakty-a-mapy	9036	2859	200	
24	1		51e134efea10ae456bb14389479ceb	http://unis.vsb.cz	0	0	!	
25	1		535a263abe76e631d6fae050403d7091	http://posta.vsb.cz	0	0	!	
26	1	2011-04-27 16:57:22.240	8d8eff8a51c9f8a6bb3ddf4fdc98e	http://www.vsb.cz/cs/okruhy/ects.html	0	120	404	
27	1	2011-04-27 16:57:23.429	49e587b84930e4f2382225c43cc8892b	http://www.vsb.cz/export_scripts/web-develop/fakulty/08.html	608	310	200	
28	1	2011-04-27 16:57:23.519	eb46e3cc146abd67aeddac942c09b	http://www.vsb.cz/export_scripts/web-develop	611	321	200	
29	1	2011-04-27 16:57:24.795	e4e9c962a8f7f24c3b250d35c5129b	http://www.vsb.cz/blog/cs/von05	5176	422	200	
30	1		dfe856fdb3386da832cb4bd1dcc1981f	http://info.vsb.cz	0	0	!	
31	1		629791a159865cf4266d853a5d944dc	http://www.vsb.cz/cs/okruhy/prezentace-univerzity/dsl	0	0		
32	1	2011-04-27 16:57:24.890	b97155d1bc608a3419a4edb188fb95	http://www.vsb.cz/cs/okruhy/management-kvality/certifikovany-system-rizeni-jakosti	0	2	600	
34	1	2011-04-27 16:57:25.852	cd954616d5cdc81c574d11e6e3ebe2b	http://www.vsb.cz/cs/okruhy/management-kvality/efqm	0	167	302	
35	1		78592a9269a64b6a217460ed529340d1	http://www.vsb.cz/cs/urceno-pro/uchazece	0	0		
36	1		f166e58a9b53732557eaa32052cd4ccd	http://www.vsb.cz/cs/urceno-pro/studenty	0	0		
37	1		dea4d97194c7af48ccd1eabb711bddf	http://www.vsb.cz/cs/urceno-pro/zamestnance	0	0	?	
38	1		b9c4a8d42cd6a67ca13268030b0565a	http://www.vsb.cz/cs/urceno-pro/absolventy	0	0	?	

Obrázek 17: Ukázka přehledu odkazů v úloze.

nepracuje.

Kliknutím na ID úlohy (1) si můžeme zobrazit odkazy nalezené v úloze. (Nyní jeden nezpracovaný bez časového razítka.) Všechny odkazy nalezené v rámci jedné úlohy lze vymazat kliknutím na ikonu bílého listu. (Zůstane pouze zadaný *startovací* odkaz.) Úlohu lze úplně odstranit pomocí ikony popelnice. **Všechny operace nad danou úlohou by měly být prováděny na zastavené úloze.**

Úlohu spustíme kliknutím na šipku.

Po překreslení stránky je indikována práce crawleru skákající postavičkou. Úlohu můžeme zastavit kliknutím na křížek (vedle popelnice a listu) v akcích. Opět si můžeme nechat rozbalit seznam odkazů kliknutím na ID úlohy, obrázek 17.

Na obrázku je zachycen stav, kdy některé stránky odkazy jsou již zpracované (obsahují číselný kód, přehled kódů je v kapitole 5.4.1), některé odkazy čekají na zpracování (kód obsahuje znak otazníku) a právě 3 odkazy jsou zpracovávány, což je vyjádřeno rotujícím kolečkem. V obrázku jsou k zahlédnutí také odkazy s vykřičníkem, které nevyhověly podmínce pro další prohledávání.

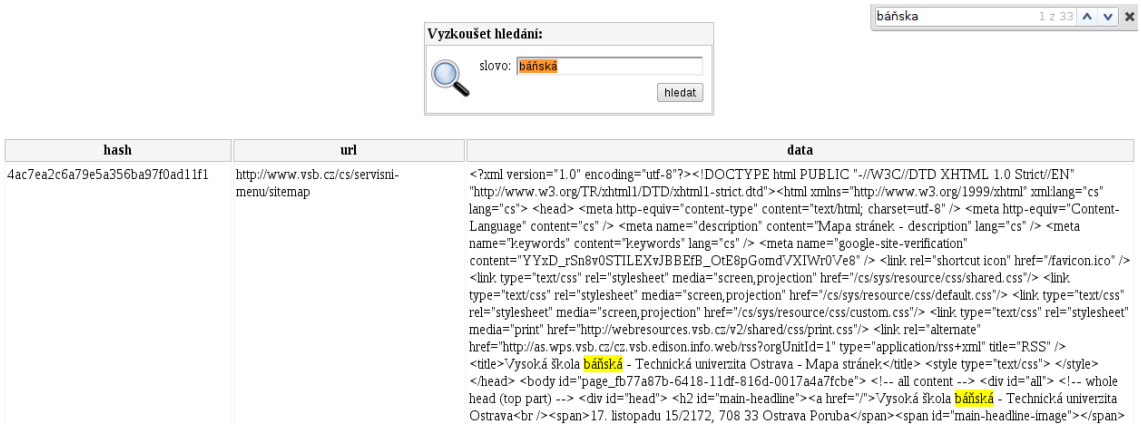
Způsob řešení aplikace nijak přesně neurčuje konec práce crawleru, např. není použito omezení hloubkou zanořeni do struktury cílového webu. Crawler pracuje dokud se mu daří nalézat nové odkazy. Parametry, které mají na množství práce pro crawler zásadní vliv, jsou dva regulární výrazy v nastavení úlohy, především pak REGEXP *omezeni pro odkazy*. Pokud již crawler nenalézá nové odkazy, přechází ke sledování expirace odkazů.

7.4 Problematika hlubokého webu

Rozsáhlou problematikou moderního Internetu je tzv. *hluboký web* (*deep web*), který je definován jako dokumenty v síti, jejichž obsah zůstává pro crawlery nedosažitelný, nikdy není prohledán a nikdy není zahrnut do indexu k možnosti vyhledávání.

Odhaduje se, že skrytý web je až 500 krát rozlehlejší, společně s informační hodnotou 1000 až 2000 krát větší, než co je dostupné v běžném (*viditelném*) webu. Tento skrytý web přesto obsahuje přes 95% veřejného obsahu a často se rozděluje podle důvodů, které brání jeho efektivnímu prohledávání prostředky crawlerů. [7] Chtěl bych zde tyto problémy zhodnotit v kontextu možností aplikace WebCrawler.

- **Soukromý web** - Považují za přirozené, že některé informace jsou na síti ukryté. Jako případní tvůrci informace máme jistě právo rozhodnout o míře veřejnosti daného dokumentu nebo datového zdroje obecně. V tomto případě je procházení obsahu jednoduše nemožné.
- **Veřejný web** - Pokud je daný dokument, resp. datový zdroj veřejně dostupný, není nijak zaručeno, že jej bude možné strojově najít a zpracovat. V tomto případě jsou největšími problémy:
  - **Vlastnický web** - Data jsou přístupná až po registraci a přihlášení. Zde záleží na úrovni propracovanosti způsobu registrace a přihlášení. Některé základní způsoby jsou jistě strojově zvládnutelné, popř. specifický účet můžeme crawleru založit i ručně. Komplikaci zpravidla představují opisy textů z obrázků. Obsah může být zpoplatněn, což sice nemusí být problémem k nalezení odkazu na dokument, ale také je třeba počítat s nutnou dohodou s poskytovatelem obsahu. WebCrawler funkčnost průchodu vlastnickým webem neobsahuje. Její implementace by náležela do oblasti vylepšení stahovače o propracovanější práci s cookies a analýzu kódu na webové formuláře.
  - **Neprůhledný web** - Je označení pro data ukrytá v hluboké struktuře provázaných dokumentů. Crawlery jsou často omezovány hloubkou zanoření jako ochranou před zacyklením v provázaných dokumentech. Součástí problému je také způsob práce se strukturou dokumentů jako se stromem, který můžeme procházet různými způsoby, tj. preferovat průchod do šířky nebo hloubky. Způsob práce ve WebCrawleru představuje průchod do šířky (po vrstvách) a hloubka není nijak omezena. Vice viz poslední odstavec kapitoly 7.3.
  - **Neviditelný web** - Do této části problému spadají stránky, na které nevede žádný odkaz. V případě HTTP můžeme crawlerem zkoušet získat adresářovou strukturu, ale to je webovými servery často zakázáno, popř. je někdy možné odhadovat pojmenování souborů dle řady jmen, kterou již známe. Dále se zde zmiňují binární formáty souborů, kde záleží na výrobci software, znalosti formátu nebo ochotě výrobce formát přepracovat k přívětivějšímu prohledávání. WebCrawler pracuje pouze s textovými daty. Implementace schopnosti zpracovat např. ZIP by zahrnovala úpravu rozhodující o další akci (rozbalení) se staženými daty před dalším zpracováním.



Obrázek 18: Ukázka vyhledávání v testovací aplikaci WebPark.

Prohledávání také komplikují způsoby návrhu moderního dynamického webu, různé načítání dat na pozadí, generování odkazů pomocí javascriptu a pod. Vypořádání se s těmito problémy často vede k návrhům specializovaných systémů k prohledávání takových prostředí, jako jsou různé vědecké encyklopedie, znalostní databáze a nebo rozsáhlé sociální sítě.

7.5 Aplikace WebPark

Testovací aplikace WebPark, která pomocí zpráv přes JMS přebírá data z aplikace WebCrawler, je dostupná na http://http://localhost:8080/WebParkEE-war/, resp. po přesměrování na http://localhost:8080/WebParkEE-war/pages/MainPage.action. Ukázka aplikace je představena na obrázku 18. Do vyhledávacího pole můžeme zadat slovo, které je pomocí jednoduchého příkazu skrze Hibernate *SELECT * FROM Page WHERE wp_t_page_data LIKE %word%;* vyhledáno ve vlastní databázi WebParku.

Aplikace WebPark má skutečně jen testovací charakter. Představuje možnosti využití různých vlastností platformy J2EE (zde je to JMS) a také naznačuje koncept, podle kterého může WebCrawler tvořit základní kámen pro rozsáhlejší aplikační řešení, kde však stále zůstává zcela zřejmé oddělení vývoje a údržby webového crawleru od dalších aplikací využívajících jeho služeb.

8 Závěr

Myslím si, že cíle práce se mi podařilo splnit. V teoretické části analýzy vybraných softwarových produktů, je možné vysledovat různorodost testovaných řešení a současně jisté společné vlastnosti nebo naopak rozdíly. Soubory nutných vlastností crawleru a také některé vlastnosti, které každý produkt činí jistým způsobem zajímavý.

Výsledkem teoretické části je především aplikace WebCrawler. Aplikaci si představuji spíše jako podnikové řešení a také jsem ji takto navrhoval. Při hledání různých informací, jsem se často setkal se definicí [1] webového crawleru jako systému, který též skladuje prohledávaná data. S touto částí definice se však úplně neztotožňuji. Jako stěžejní cíl práce WebCrawleru jsem sledoval sběr odkazů na místa do internetu, popř. podnikové sítě. Jistě existují případy užití, kdy se bez skladování dat neobejdeme, ale též si lze představit, že nám stačí data stáhnout a jen nějak zpracovat. Hledání neoprávněného užití textů a citací, škodlivého kódu, software pro hledání podobností mezi obrázky a pod. V takových případech nemusí být nutné (nebo dokonce praktické) data skladovat, přesto potřebujeme aplikaci pro systematické procházení míst v síti. - Webový crawler.

Výše uvedenou představu jsem se pokusil podpořit implementací malé aplikace WebPark, která data z WebCrawleru sice skladuje, avšak tvoří zcela oddělenou část celého řešení, viz závěr kapitoly 7.5.

V aplikaci existují jistě nedostatky a oblasti, které si zaslouží pozornost a případné vylepšení. V oblasti GUI je vážným nedostatkem chybějící stránkování v seznamu odkazů. Stahovač nepracuje s HTTPS a neumožňuje práci přes proxy. Parser odkazů nerozlišuje různé části stránek a hledá parametry s odkazy bez ohledu na to, zda se jedná o část HEAD, BODY a vyhledá též odkazy v komentářích. To může být problém. Užitečnou vlastností by byla také schopnost práce s cookies.

9 Reference

[1] KRITIKOPOULOS, Apostolos - SIDERH, Martha - STROGGILOS, Kostantinos. *CrawlWave: A Distributed Crawler* [online]. 2004 [cit. 2011-04-25]. URL: <http://aiw.cs.aueb.gr/pub/crawlwave.pdf>

[2] *The Web Robots Pages* [online]. 2007, 2010-08-23 [cit. 2011-03-26]. URL: <http://www.robotstxt.org/robotstxt.html>

[3] PUŽMANOVÁ, Rita. *TCP/IP v kostce*. 1. vyd. České Budějovice: KOPP, 2004. 607 s. ISBN 80-7232-236-2.

[4] *CZ.NIC - IDN - Internationalized domain names* [online]. 2011, [cit. 2011-03-24]. URL: <http://www.xn–hkyrky-ptac70bc.cz/>, <http://www.háčkyčárky.cz/>

[5] *ICANN | Internationalized Domain Names Glossary* [online]. 2011, 2010-08-13 [cit. 2011-03-24] URL: <http://www.icann.org/en/topics/idn/idn-glossary.htm>

[6] MAGED Michael, MOREIRA E. José, SHILOACH Doron, WISNIEWSKI W. Robert. *Scale-up x Scale-out: A Case Study using Nutch/Lucene* [online]. 2007-02-14 [cit. 2011-02-20]. URL: <http://www.cecs.uci.edu/ papers/ipdps07/pdfs/SMTPS-201-paper-1.pdf>

[7] MathAn Praha, s.r.o. *Neviditelný web - Infogram* [online]. 2011, [cit. 2011-05-01]. URL:<http://www.infogram.cz/article.do?articleId=1765>

[8] Net Communities, Ltd. *Google Mini 2.0 review - IT Reviews* [online]. 2006-11-28 [cit. 2011-04-31]. URL: <http://www.cecs.uci.edu/ papers/ipdps07/pdfs/SMTPS-201-paper-1.pdf>

10 Přílohy

K bakalářské práci je přiloženo CD s tímto obsahem:

1. PDF verze této bakalářské práce. (Není součástí přílohy v IS.)
2. Aplikační archívy EAR pro deploy do aplikačního serveru Glassfish2. (Z důvodu úspory místa nejsou archívy součástí přílohy v IS, je možné je získat překladem ze zdrojových kódů.)
3. Zdrojové kódy aplikací WebCrawler (WebCrawlerEE) a WebPark (WebParkEE) ve formě projektů pro vývojové prostředí NetBeans IDE 6.9.1
4. Skripty pro databázový server MySQL k vytvoření databázových schémat a přístupů pro aplikace WebCrawler (WebCrawlerDB.sql) a WebPark (WebParkDB.sql).